

## **Chapter 1**

### **Introduction**

ICKY is a multiple resolution video card for the Macintosh II PDS computers. Supported monitors include the Macintosh 13 inch, the Macintosh 12 inch, and the Bartlett internal SE/30 gray scale adapter. The card supports 1, 2, 4, 8, and 24 bit mode. Virtual video modes are supported, which allows large virtual screens to be displayed via hardware pan. The Maverick chip is used to do all PDS, VRAM, and Video timing.

## Chapter 2

### Macintosh II PDS Family

The Macintosh SE/30 contains the Motorola 68030 microprocessor and one expansion slot, which may be numbered 9, A, or B hex. The SE/30 is essentially a Macintosh IIx computer in a smaller box. The expansion slot, though, comes off of the processor bus. The SE/30 does not have any NuBus expansion slots.

While the Macintosh SE/30 has 32 address lines, only 24 of these are used in the normal (default) mode. The computer can be placed in 24-bit or 32-bit mode similar to the Macintosh II. The 8-bit video board's ROM and VRAM array addresses, however, are all directly accessible in normal mode.

Address	Description
00000000 – 000FFFFF	RAM (minimum configuration)
00100000 – 00CFFFFF	RAM (expansion area)
00D00000 – 3FFFFFFF	RAM (undefined)
40000000 – 4007FFFF	ROM Bank 0 (minimum configuration)
40080000 – 4FFFFFFF	ROM (undefined)
50000000 – 50001FFF	VIA1 (x0200)
50002000 – 50003FFF	VIA2 (x0200)
50004000 – 50005FFF	SCC (x0002)
50006000 – 50007FFF	SCSI (Handshake)
50010000 – 50011FFF	SCSI (x0010)
50012000 – 50013FFF	SCSI (Pseudo DMA)
50014000 – 50015FFF	Sound
50016000 – 50017FFF	SWIM
50018000 – 57FFFFFF	(undefined)
58000000 – 5FFFFFFF	030 Direct Slot expansion (if pseudo-NuBus is not used)
60000000 – F8FFFFFF	expansion (undefined)
F9000000 – FBFFFFFF	expansion pseudo-NuBus slots
FC000000 – FDFFFFFF	expansion (undefined)
FE000000 – FE00FFFF	video RAM space
FEFF0000 – FEFFFFFF	video ROM space
FF000000 – FFFFFFFF	expansion (undefined)

Table 2-1 Address Mapping

The Macintosh SE/30 uses memory-mapped I/O. Each device in the system can be accessed by reading or writing to specific address locations in the address space of the computer. The addressing for a card in a slot is directly dependent on the slot number. The SE/30 has three slot numbers possible for a single slot. The slot number is defined in hardware on the expansion board and cannot be changed.

The slot in the Macintosh SE/30 can be directly accessed in the 32-bit mode at the address found by multiplying the slot ID by 10,000,000 hex (s000 0000-sFFFF FFFF hex); this is the Super Slot Space. The slot can also be found at the Slot Space Fs00 0000-FsFF FFFF hex (s = slot ID, 9-B) when in the 24-bit default operating mode. Each physical slot has both the slot and super slot space allocated to it. In 24-bit mode the AMU (Address Mapping Unit) or PMMU (Paged Memory Management Unit), whichever is present, will convert s0 0000 to Fs00 0000.

## 2.1 Bus Signals

The Macintosh SE/30 Direct Slot expansion bus is based on the Motorola 68030 microprocessor. It is machine-specific, not a general-purpose bus like NuBus. The card design may be similar to NuBus (Pseudo-NuBus), with the same type of ROM and addressing, but the pinout and signals differ. The SE/30 Direct Slot uses a 120-pin 16MHz synchronous bus. Words, halfwords, and bytes can be read and written, but the bus is optimized for word transfers. All signals are active low.

Bit 31	Bit 0		
Direct Slot Word			
Halfword 1	Halfword 0		
Byte 3	Byte 2	Byte 1	Byte 0

Table 2-2 Bus Data

The bus specification divides the signals into five types:

Power: +5V, +12V, -12V, and -5V.

Data and Address Lines: A0-A31, D0-D31.

Control Lines: \STERM, SIZ0-SIZ1, FC0-FC2, \RESET, \BERR, R/W, \AS

Clocks: CPUclock and C16M (for the SE/30, these are identical 16 MHz signals).

Machine-Specific Signals: PWROFF, \BUSLOCK, \IRQ1-\IRQ3, \TM0A-\TM1A, \NuBus, CPUclock.

The connector contains three rows of pins (see Appendix B). The Micron video board does not use all of the signals on the expansion bus. A 'high' level is > 2V, 'low' level < 0.8V.

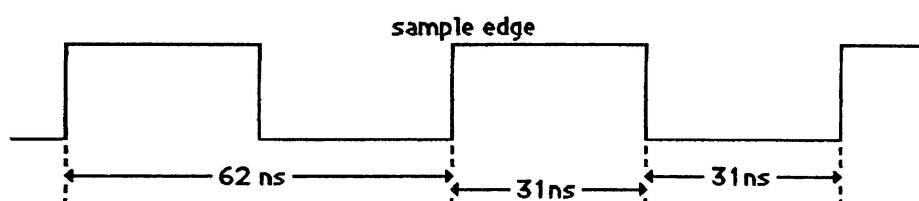


Figure 2-1 System Clock

## **2.2 READ-WRITE-READ CYCLE**

The read-write-read transaction is shown below. It may take longer than shown, but the timer on the SE/30 main logic board will generate a bus error signal when the address strobe is asserted for longer than  $\approx$ 20 microseconds.

### **SE/30 Read/Write Cycles**

(All accesses to the video card are 32 Bit Synchronous)

#### **Read Cycle**

At state zero (S0) the SE/30 drives Ax, FCx, SI<sub>Z</sub>x, and R\W with an address, type of function, size of data transfer, and read command. Half a clock cycle later, during state one (S1), \AS and \DS are asserted by the SE/30 to tell the external device that the address lines are now valid and data may be placed on the bus. By the end of S1 the external device will assert \STERM. \STERM tells the SE/30 that the read data will be valid by the next falling clock. At the start of state two (S2) the external device places data on the bus and terminates \STERM. The SE/30 latches in the data at the end of S2. State three (S3) provides the data hold time as required.

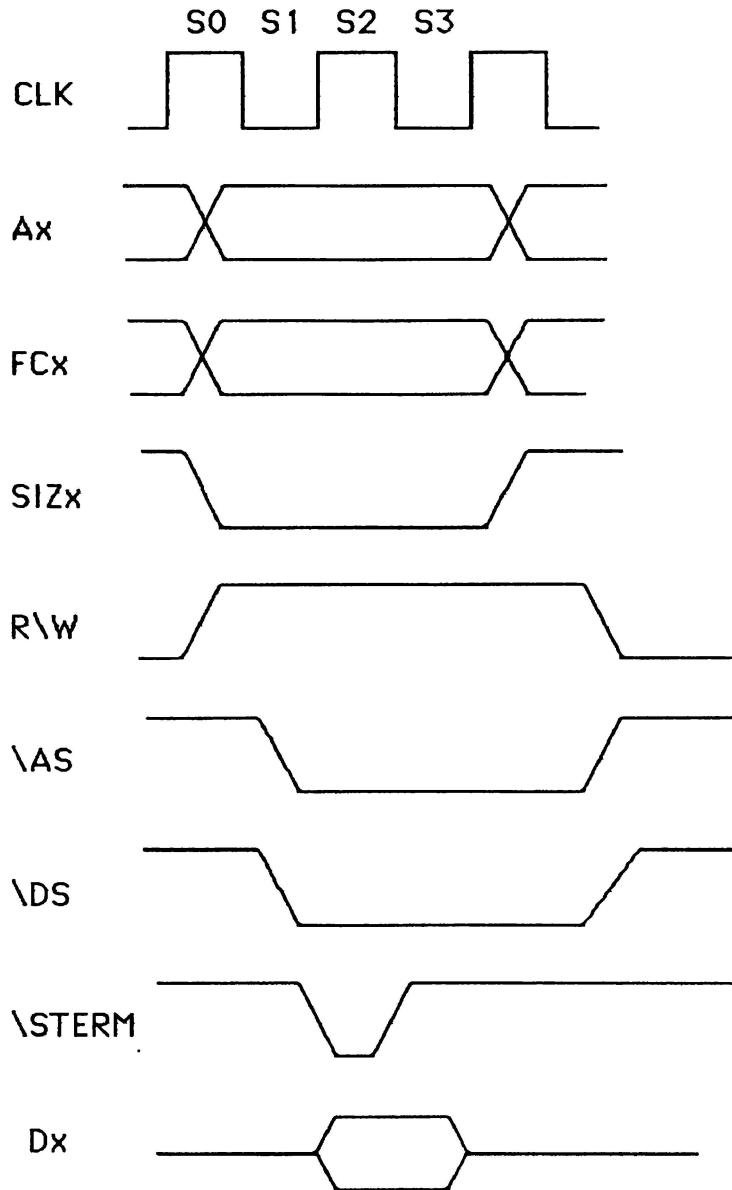


Figure 2-2 SE/30 32 Bit Synchronous Read

### Write Cycle

At state zero (S0) the SE/30 drives Ax, FCx, SIZx, and R\W with an address, type of function, size of data transfer, and write command. Half a clock cycle later, during state one (S1), \AS is asserted by the SE/30 to tell the external device that the address lines are valid. At the beginning of state two (S2) the SE/30 looks for \STERM. If \STERM is not present the SE/30 will insert a wait state. Also during S2 the SE/30 will drive the data bus with valid data. During a wait state (SW) the SE/30 asserts \DS to tell the external device that the bus still contains valid data. At the start of each wait state the SE/30 will check for \STERM. If \STERM is not present the SE/30 will insert another wait state, if

\STERM is present, the SE/30 will terminate the write cycle in the following state (S3). During state three (S3) the address and data lines remain valid.

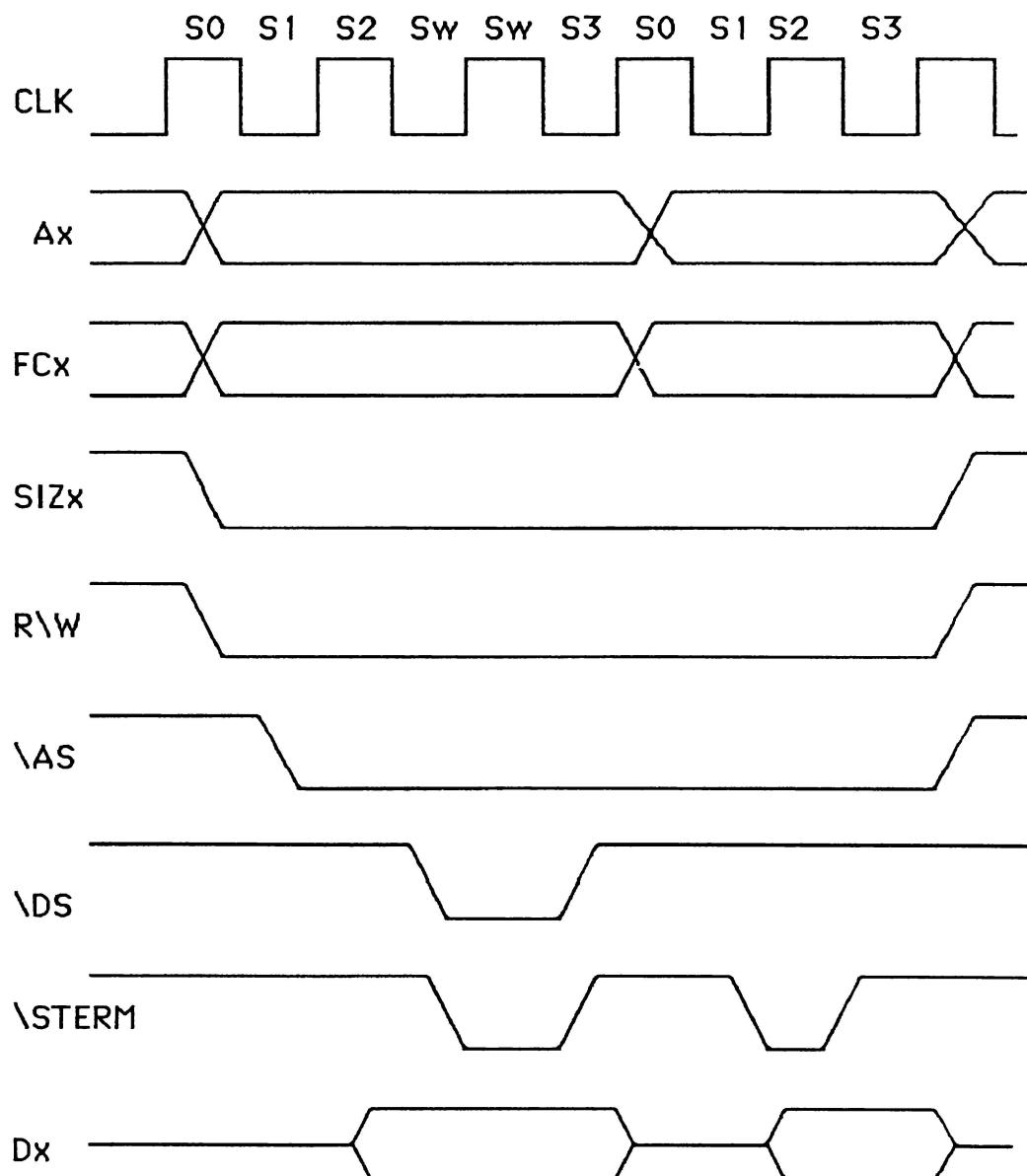


Figure 2-3 Read-Write-Read Cycles

## 2.3 System Board

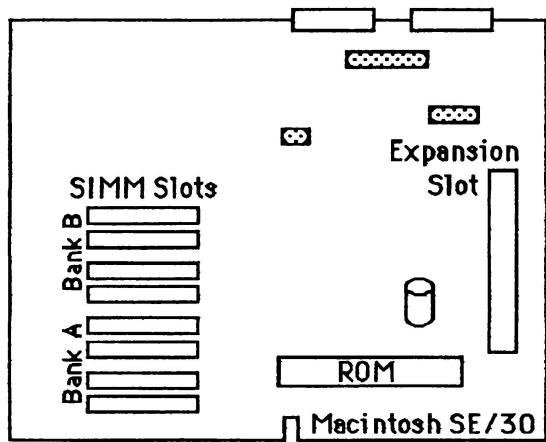


Figure 2-4 Macintosh SE/30 System Board

## **Chapter 3**

### **Theory of Operation**

**3.1 PDS Interface**

**3.2 Video Ram Control**

**3.3 Pixel Data Formatting**

**3.4 RGB Video Output**

### **3.1 PDS Interface**

The BIVT controls the operation of the PDS interface. Four 74F245s are used to buffer the data and three 74ALS373s are used to latch the address.

For VRAM access the BIVT will generate all timing for RAS, CAS, row and column muxing. Only one RAS is used on Icky. Two CASs are used to support the 2 way interleaved memory. The BIVT will hold off access to the VRAM if a VRAM transfer cycle is needed.

For ROM reads the BIVT will generate the ROMCS signal.

For DAC access, the BIVT will generate DACRD and DACWT signals. DAC access will be held off during active video if the NOSNOW bit is enabled within the BIVT.

Also the BIVT will accept writes to its internal registers. These are used to control the operation and configuration of the BIVT.

### **3.2 Video Ram Control**

All video ram control except SC0 and SC1 is provided by the BIVT. For bus access, the row and column address are muxed in three external 74F257s. For transfer cycles the BIVT generates the row and column addresses. Refresh is also generated by the BIVT.

### **3.3 Pixel Data Formatting**

The VRAM is organized as two 32 bit wide interleaved banks. The two banks are capable of providing 4096 bytes per row. The pixel data is clocked out of the VRAMS by two out of phase clocks SC0 and SC1. The data then enters a 64 to 32 bit mux comprised of 8 74F257s. The mux alternates between the two banks and is controlled by PDBSEL. After the mux, the data can take one of two paths depending on if it is in 24 or 8 bit mode. Also note 1, 2, and 4 bit modes are just processed 8 bit mode. The first path is for 24 bit mode. Here the data bits 8 through 23 pass directly to the DAC. Bits 24 through 31 are ignored and bits 0 through 7 are passed through a mux and on to the DAC.

In 8 bit mode, the pixel data is routed through a 32 to 8 bit mux. The mux is comprised of 4 74F253s. PBSEL0 and PBSEL1 control which one of four bytes is selected. After the mux, the data is registered in a 74F374 to provide enough setup time for the pixel mixer pal. The pixel mixer pal will then format and shift the data out depending which mode is selected. The output of the pixel mixer is passed through a mux on to the DAC.

U25 generates all signals to control the muxes, shift clocks, and pixel mixer pal. FORRUN is the signal generated in the BIVT which signals when to start a new video line running.

### 3.4 DAC

The pixel data is converted to an analog voltage using a 473 DAC (U45). This DAC is capable of accepting either 24 bits of data for true color operation, or 8 bits for pseudo color. Also when in 8 bit mode, individual bits can be masked off, thus allowing support for 1, 2, and 4 bit modes. The selection of 8 or 24 bit mode is accomplished with the mode2 signal.

Along with the pixel data, sync and blank signals are also supplied to the DAC. These two signals are generated in the BIVT asic.

The BIVT generates the DAC read and DAC write signals which read and write the control registers inside the 473 DAC.

Figure 3-10 Video Generation Block Diagram

The following diagram illustrates the operation of a video D/A. In practice the blanking and sync intervals will be longer.

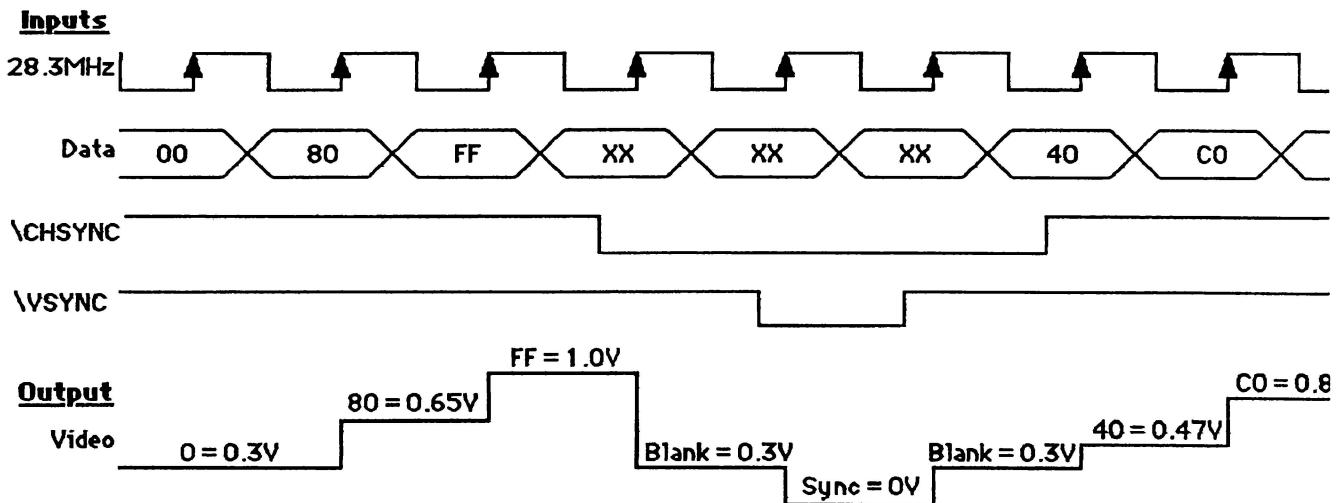


Figure 3-11 Video D/A Operation

More complete and detailed video diagrams can be found in the Electronic Industries Association specification EIA-343-A.

## Appendix C

### Maverick Theory of Operation

#### Introduction:

The Maverick gate array is designed for use on Macintosh video boards. It includes both SE/30 and NuBus bus interface circuits, and video timing generation circuits.

There are currently three different bond options for the chip. All inputs and outputs are available in the 'Top Gun' 120-pin option. NuBus-specific signals are not present on 'Goose' which is an SE/30 only 100-pin option. SE/30 specific interface signals are not present on 'Maverick' which is a 100-pin NuBus only option. It is expected that the 120-pin option will not be used in production unless problems are encountered in one or both of the 100-pin options (insufficient power and ground pins, noise, etc.). For the purposes of this document all versions of the chip will be referred to by the common name Maverick.

#### Bus Interface:

The bus interface includes address decoding for ROM, VRAM, DAC, internal mode registers and external mode registers.

ROM accesses are decoded for the top two megabytes of the slot address space. A ROMCS strobe of approximately 250 nanoseconds will be generated to enable the ROM data to the data bus unless a VRAM transfer cycle is being performed. If a VRAM transfer cycle is being performed the ROM cycle will be held off until the transfer is complete, and then the ROM cycle will occur.

DAC accesses are decoded for the next highest two megabytes of address space. There are two modes for accessing the DAC. If the G300EN mode bit is low, the chip will assume that a G300 is present. In this case the DAC access is synchronized to the pixel clock to guarantee proper setup and hold times. The DACCS signal will go low followed immediately by the DACWRT signal (for processor write cycles). The DAC cycle will last approximately eight pixel clock cycles with the DACWRT signal going high before the DACCS signal goes high. If the G300EN mode bit is high, the chip will assume that the DAC is not a G300. In this case the DACCS signal is forced low for DAC read cycles, and the DACWRT signal is forced low for DAC write cycles. Again if a VRAM transfer cycle has begun when a DAC access is initiated, the cycle will be held off until the transfer is complete.

DAC accesses may also be held off until horizontal or vertical video blanking intervals if the NOSNOW ENABLE mode bit is high. This prevents DAC access cycles from interfering with the video display. For the NuBus interface the DAC cycle will be held off until a blanking interval occurs. For the SE/30, the BERR

and HALT bus control lines are both driven low which instructs the processor to retry the bus cycle. This prevents a bus time out error from occurring. There is no method of signaling the NuBus to retry the bus cycle, so if the active line time is longer than the NuBus time out period, the NOSNOW ENABLE will have to be written to a low to disable this feature.

The next lower four megabytes of address space are decoded as mode accesses. There are internal mode registers, external mode bits, and external mode read and write control lines (XMODERDB and XMODEWRTB). An address map is included which lists the address map for the specific mode bits.

The bottom eight megabytes are decoded as VRAM space. There are three mode bits that determine the VRAM bank configuration. One, two or four column banks are allowed; and two row banks are allowed which may be either two or four megabytes per bank. Two RAS strobes, four CAS strobes, four write enables and one TROE strobe are generated. VRAM bus cycles are also held off for VRAM transfer cycles. There is also a NuBus block mode transfer enable bit which is valid only for VRAM accesses.

The BUSTYPE input pin is used to select which bus interface will be used. This pin is pulled down on the chip. On the 120 pin package it will be pulled up externally for SE/30 direct slot interfaces, and pulled down for NuBus. On the SE/30 100 pin package it will be pulled up, and on the NuBus 100 pin package it is not available as a pin (it is internally pulled down).

#### Vertical Blanking Interrupt:

On Macintosh video boards there is an interrupt generated at the video vertical retrace rate. The interrupt is asserted at the beginning of the vertical blanking period and is held until it is cleared out by the interrupt routine in the video driver. The interrupt is cleared out by writing any data to the mode register at address offset A00014 into the slot space. This corresponds to a dummy mode register write.

#### External Mode Bits:

Four general purpose external mode bits are provided (XEXTMODE(0-3)). These are written to in the same manner as the internal mode bits, but have no internal function. They are intended to be used as pixel depth mode bits, and possibly as video frequency select lines.

There is also a pin called DEBUG which is the output of a four to one multiplexer. The inputs to the mux are TRANSWAITB, DRAM, CYCLE1, and ModesB(3). The first three are signals which may be useful for chip test and debug. ModesB(3) may be used as a general purpose external mode bit. Two other internal mode bits select which of the four signals will go off-chip.

#### VRAM Transfer Cycles:

There are several different VRAM transfer modes supported.

A programmable number of refresh cycles are performed before each transfer cycle. The number of refresh cycles required per transfer is dependent on the actual transfer cycle frequency which will vary with display resolution, and with the particular transfer cycle mode. Maverick also drives the RAS, CAS, TROE, and DSF signals for the transfer cycle. If a bus cycle is in progress when a transfer cycle request is generated, the bus cycle will complete before the refresh and transfer cycles begin.

If the G300EN bit is low the XTREQ pin is driven from the G300 bus request pin which signals Maverick that it is time to perform a transfer cycle. Maverick will wait until any current bus cycles are complete, and then will assert the XVXAMUXENB signal to signify that the G300 may drive the transfer address. The G300 will then drive the Maverick XTROEIN signal which is used by Maverick to generate the TROE signal for the VRAMs.

If the G300EN bit is high and the Split and Seamless bits are low, then a transfer cycle request will occur with each horizontal sync pulse.

If the G300EN bit is high, the Seamless bit is low and the Split mode bit is high, then the XTREQ input pin is driven from the QSF pin of the VRAMs. In this mode a full transfer cycle is performed at the beginning of each vertical sync, followed by a split transfer at the end of the vertical sync. Then a split transfer will be performed on each transition of the XTREQ pin.

If the G300EN bit is high and the Seamless bit is also high, then Maverick will perform seamless transfers at a rate determined by the TRANSREQ and REQTRANS mode registers. Once a transfer cycle has been performed, the TRANSREQ register is counted down and then another transfer cycle request is performed. Once the transfer cycle request is acknowledged, a programmable number of refresh cycles are performed and the REQTRANS register is counted down. When the refresh cycles are complete, TROE is driven low, and when the REQTRANS count is reached, TROE is driven back high to perform the transfer. At this time the TRANSREQ count is reloaded and the cycle is repeated.

The transfer address for the VRAMs can be generated in Maverick. The transfer address format is determined by three mode bits (see Table 1). The different formats are required to support one or four meg VRAMs, and to provide flexibility in the effective definition of the row x column dimensions. The initial row and column addresses are written into the mode registers. The row address is then incremented with each transfer cycle provided that the zoom factor is set to zero. For zoom factors other than zero the zoom count will decrement with each transfer and the address will be incremented when the zoom count reaches zero. The zoom count will then reload and begin counting out the next group of transfer cycles. This allows for a programmable one-dimensional zoom in the 'Y' axis. 'X' axis zoom must be accomplished by pixel replication which will be a function of the pixel clock and shift clock generation circuitry.

Mode Bits				
A5	A4	A3	Row	Column

0	0	0	R0-R8	C0-C8
0	0	1	R1-R9	C0-C7,R0
0	1	0	R2-R10	C0-C6,R0-R1
0	1	1	R3-R11	C0-C5,R0-R2
1	0	0	R0-R7(R8)	C0-C7,(R0)
1	0	1	R1-R8(R9)	C0-C6,R0,(R1)
1	1	0	R2-R9(R10)	C0-C5,R0-R1,(R2)
1	1	1	invalid mode	invalid mode

Table 1

Mode bit A5 is intended to be set high for 256K VRAMs. For 256K VRAMs the addresses in ()'s will not be used.

#### Video Timing Generator:

Video timing signals are generated by counting through a series of mode registers associated with the vertical and horizontal video waveforms. Please refer to the attached drawing for a pictorial description of the timing elements. The horizontal state machine cycles through each element twice per line to allow for equalization pulse generation.

Most of the timing elements will be self-explanatory to those familiar with video timing signals; however, there are some timing elements that are unique to the Maverick design. The FS1 period is used to fill any pipeline that may exist between the VRAM and the DAC. The FS2 period is used to empty the pipeline. NS defines the period where bus access to the DAC will not interfere with the generation of video. HS22 is used for the generation of equalization pulses only.

Interlaced and noninterlaced video formats are supported. Vertical state transitions may be made at half-line or whole-line boundaries. Equalization pulses in the vertical blanking signal may be enabled or disabled. Display sizes up to 2K x 2K are supported. Vertical sync and field ID may be generated externally or internally. For external vertical sync Maverick will wait for an external sync pulse at each vertical front porch period. The falling edge of the active low external sync will force the vertical timing state machine to go into the internal vertical sync state. NTSC timing waveforms are generated by putting the video timing in interlaced mode, selecting the proper horizontal and vertical resolutions, using half line counts in the vertical state machine, and turning on the equalization pulses.

```
GBLA    &DriverOnChip
&DriverOnChip ,`SETA    1
;-----  
;  
;369Color.a  
;DESCRIPTION:  
;    This is the driver file for video board 369 XCEED SE/306-48.  
;    Code Name: Bart  
;    Product Name: Micron XCEED Color 30™  
;MAKEINFO:  
;    Run BinHex, choose "Application->Upload" on the 369Color.d file ~  
;    and name the file 369Color.text.  
;        asm 369Color.a  
;        link 369Color.a.o -ss 64400 -o 369Color.crc  
;        crcPatch 369Color.crc  
;        Data 369Color.crc 369Color.d
;VERSION:  
;    Version 1.2      August 22, 1990 Laura Lorraine Shaffer Mills  
;    Version 1.3a3    January 2, 1991 Tom Stamm  
;    Copyright Micron Technology, Inc. 1990/1991.  
;    All Rights Reserved.
;HISTORY:  
;    A0.2    cb      (02/06/88) changed _DoVBLTask to indirect jump thru jVBLTask.  
;    A0.3    cb      Indexed SetEntries/GetEntries Fixed.  
;    A0.4    cb      Fixed Gamma and Grayscale.  
;    080089  cb      Created.  
;    061590  llsm     Changed version number from 1.0 to 1.1 (PageMaker fix).  
;    082290  llsm     Changed Rev. 1.1 to 1.2; ROM revision from 1 to 2.  
;    120390  ts      Change Rev. 1.2 to 1.3a1; Add changes for Maverick Chip. ~  
;                    Change the start address offset for each screen depth.  
;    121290  ts      Put in register transformations for Maverick.  
;    121390  ts      Put in new register transformations for Maverick.  
;    121490  ts      Add logic in primary init and add secondary init ~  
;                    for the smiley mac.  
;    121890  ts      Add resources for Virtual Video.  
;    010291  ts      Set the offset to the next row down for virtual ~  
;                    video.  
;    011591  ts      Add resources for VGA, Portrait and Bartlett  
;    062091  ts      Add new portrait numbers for Maverick.  
;    112591  llsm     Changed board number 281 to 369. Added Hungarian notation.  
;    112691  llsm     Changed ROM revision level from 2 to 3; made constant in ~  
;                    the include file.  
;    112791  llsm     Put the clock into the External register.  
;    010292  llsm     Changed from 64K to 256K ROM.  
;    011492  llsm     Change from Maverick to Gambler, with G300 bit (in ModesA ~  
;                    register) inverted.  
;    011692  llsm     The Debug Select bits in ModesB register changed from ~  
;                    3 to 0.  
;    052793  rb      The start of the HR series after 2 months of  
;                    work. Isn't life grand??  
;                    Check separate notebook for changes.
```

;  
;NOTES:  
; 1. Released original revision 1.  
; June 15, 1990 Released revision 1 (PageMaker).  
; August 22, 1990 Released revision 2 (SetInterrupt).  
;  
; 2. HARDWARE/SOFTWARE INTERFACE INFORMATION  
;  
; MODE REGISTER is at address sE0000 (write-only)  
; bits 24:25 set 1, 2, 4, or 8 bit mode (values 0, 1, 2, 3 respectively)  
; bit 26 1 enables interrupts, 0 disables interrupts  
; bit 27 1 clears interrupts disregarding bits 0:2  
; 0 clears interrupts and uses information in bits 0:2  
;  
; CLUT WRITE ADDRESS REGISTER is at address sC0000 (write-only)  
; bits 0:7 set the starting CLUT entry to write (0-255)  
; subsequent writes to the CLUT DATA REGISTER set the RGB component  
; values in the CLUT table starting at the entry specified.  
;  
; CLUT READ ADDRESS REGISTER is at address sC000C (write-only)  
; bits 0:7 set the starting CLUT entry to read from (0-255)  
; subsequent reads of the CLUT DATA REGISTER get the RGB component  
; values in the CLUT table starting at the entry specified.  
;  
; CLUT DATA REGISTER is at address sC0004 (read/write)  
; bits 0:7 contain the RGB component data. The first read/write acts on the  
; red component, followed by the green and then the blue.  
; Subsequent accesses get/set data for the next CLUT entry, etc.  
;  
; CLUT ADDRESS MASK REGISTER is at address sC0008 (write-only)  
; bits 0:7 select the address lines valid for the current mode selection  
; use \$01, \$03, \$0F, \$FF for 1, 2, 4, and 8 bit modes respectively  
;  
; 3. To use the 32K ROM, put this code in the last 1/4 of the ROM -- ~  
; fill the rest with F's. The code is ~  
; DCB.B (24\*1024)-(\*-VideoDeclROM), \$FF.  
;  
; 4. The sIZER program has an error where it writes into vendorUse4 when ~

```

; it should be in vendorUse3 (the only time vendorUse3 should be ~
; written is when the monitor is changed by the user by the Monitor -
; Extension or sIZER). I'm saving 4 into 6 as well so I can -
; determine when it gets screwed up.
-----
TITLE          'Board 369 Driver (XCEED SE/306-48)'
OPT           ALL
STRING         C
MACHINE        MC68020
CASE           OFF
BLANKS          ON
PRINT          OFF

INCLUDE 'QuickEqu.a'
INCLUDE 'SysEqu.a'
INCLUDE 'ROMEqu.a'
INCLUDE 'Traps.a'
INCLUDE 'Syserr.a'
INCLUDE 'VideoEqu.a'
INCLUDE 'SlotEqu.A'

MACRO
MyRevLev
DC.L          'Rev. 1.1', 0           ;llsm 6/15/90
DC.L          'Rev. 1.2', 0           ;llsm 8/22/90
DC.L          'Rev. 1.3a3', 0          ;ts 12/18/90
DC.L          'Rev. 2.0', 0            ;rb 5/21/93
ENDM

MACRO
MyBoard
DC.L          'XCEED Color 30HR™ V2.0', 0
ENDM

PRINT          ON

INCLUDE '369includes.a'
-----
; MAIN
; -----
VideoDeclROM   MAIN

; To use the 32K ROM, put this stuff in the last 1/4 -- fill the rest with F's.
; The code is DCB.B (24*1024)-(*-VideoDeclROM), $FF
;rb NOTE 7
DCB.B          (16*1024)-(*-VideoDeclROM), $FF ;rb NOTE 7

_sRsrcDir      OSLstEntry.      _sRsrcBoard
                _sRsrc6x4Video, _sRsrc6x4Video      ; 24-bit 6x4 monitor
OSLstEntry      _sRsrcVGA, _sRsrcVGA      ; 24-bit 6x4 VGA
OSLstEntry      _sRsrcPortrait, _sRsrcPortrait      ; 24-bit Portrait
OSLstEntry      _sRsrcBartlett, _sRsrcBartlett      ; 24-bit Bartlett (512x342)
OSLstEntry      _sRsrcVV1kx1k, _sRsrcVV1kx1k      ; 24-bit 1kx1k monitor
OSLstEntry      _sRsrcVV1kx512, _sRsrcVV1kx512      ; 24-bit 1kx512 monitor
OSLstEntry      _sRsrcctwelve, _sRsrcctwelve      ; r.b. new rez svga
OSLstEntry      _sRsrcfifty, _sRsrc8x6      ;r.b. NOTE 28 new rez svga
OSLstEntry      _sRsrcsixty, _sRsrc8x6      ;r.b. NOTE 28 new rez svga
OSLstEntry      _sRsrcseventy, _sRsrc8x6      ;r.b. NOTE 28 new rez svga
OSLstEntry      _sRsrc832x624, _sRsrc832x624      ;r.b. new rez apple 16inch
OSLstEntry      _sRsrc1x7, _sRsrc1x7      ;rb NOTE 28
OSLstEntry      _sRsrc6x4Video32, _sRsrc6x4Video32      ; 32-bit 6x4 monitor
OSLstEntry      _sRsrcVGA32, _sRsrcVGA32      ; 32-bit 6x4 VGA
OSLstEntry      _sRsrcPortrait32, _sRsrcPortrait32      ; 32-bit Portrait
OSLstEntry      _sRsrcBartlett32, _sRsrcBartlett32      ; 32-bit Bartlett (512x342)
OSLstEntry      _sRsrcVV1kx1k32, _sRsrcVV1kx1k32      ; 32-bit 1kx1k monitor
OSLstEntry      _sRsrcVV1kx51232, _sRsrcVV1kx51232      ; 32-bit 1kx512 monitor
OSLstEntry      _sRsrcctwelveQ32, _sRsrcctwelveQ32      ;r.b. new rez svga
OSLstEntry      _sRsrcfiftyQ32, _sRsrc8x6Q32      ;r.b. new rez svga
OSLstEntry      _sRsrcsixtyQ32, _sRsrc8x6Q32      ;r.b. new rez svga
OSLstEntry      _sRsrcseventyQ32, _sRsrc8x6Q32      ;r.b. new rez svga
OSLstEntry      _sRsrc832x624Q32, _sRsrc832x624Q32      ;r.b. new rez apple 16inch
OSLstEntry      _sRsrc1x7Q32, _sRsrc1x7Q32      ;rb NOTE 28
DatLstEntry     EndOfList, 0

_sRsrcBoard     OSLstEntry.      _sRsrcType, _BoardType
                _sRsrcName, _BoardName
OSLstEntry      _sRsrcIcon, _BoardIcon
DatLstEntry     BoardID, kColor8BoardID
OSLstEntry      PrimaryInit, _sPInitRec
OSLstEntry      VendorInfo, _VendorInfo
OSLstEntry      SecondaryInit, _sSInitRec
DatLstEntry     EndOfList, 0

_BoardType      DC.W          catBoard, typBoard, 0, 0
_BoardName      DC.W          myBoard
_BoardIcon      DC.W          $FFFF,$FFFF,$FFFF,$FFFF,$FFFF,$FFFF,$FFFF
                    $FFFF,$FFFF,$BFA3,$E3FF,$9FC3,$C3F9,$8FEB,$CF9

```

```

DC.W      $87F3,$CFE1,$A3FB,$FFE3,$81FF,$FF81,$88FF,$FF89
DC.W      $807F,$FE01,$A23F,$FE23,$801F,$F801,$888F,$F889
DC.W      $800F,$F001,$A23F,$FA23,$803F,$FC01,$88FF,$FE89
DC.W      $80FF,$FF01,$A3FF,$FFA3,$83FB,$DFC1,$8FFB,$CFE9
DC.W      $8FE3,$C7F1,$BF3,$E3FB,$BF83,$C1FD,$FF8B,$C8FF
DC.W      $FE03,$C07F,$FE23,$E23F,$F803,$C01F,$FFFF,$FFFF

;-----
;PrimaryInit
;DESCRIPTION:
; PrimaryInit _sPInitRec is an sExecBlock (C&D 8-3).
;DECLARATION:
; _sPInitProc;
;ARGUMENTS:
; On Entry    A0 = Ptr to SEBlock.
; On Exit     seStatus field = +1 if successful init.
;             seStatus field = -1 if error during init.
;EXAMPLE:
; N/A.
;HISTORY:
; 112691 llsm Named registers.
; 112791 llsm Put clock value in External register.
; 120291 llsm Changed all ExtMode and ModesB Maverick values. The IRE ~
;               bit moved from ExtMode to ModesB. ExtMode now has the ~
;               monitor and depth (1-bit mode).
; 120691 llsm Can exit with error if no cable or Bartlett.
; 121191 llsm Changed Maverick numbers. Color table was set wrong, was ~
;               using ClutReadOffs instead of ClutDataRegOffs.
; 121291 llsm Write vendorUse4 into vendorUse6 as well, to fix sIZER ~
;               error (writes to 4 instead of 3). If 4 and 6 are ~
;               different, then move 4 to 3 and 6 to 4.
; 121391 llsm Changed the Maverick numbers for Portrait to remove white ~
;               line on right when Gamma Contrast was turned down.
; 121791 llsm Changed Maverick numbers for everything -- Rev. F?
; 121891 llsm Changed Maverick numbers.
; 121991 llsm Changed Maverick numbers for Bartlett.
; 122091 llsm Put VFPOrch value for 6x4 virtual here. Was in SetMode, ~
;               but doesn't depend on mode.
; 011492 llsm Change from Maverick to Gambler, with G300 bit (in ModesA ~
;               register) inverted.
; 011692 llsm The Debug Select bits in ModesB register changed from ~
;               3 to 0 ($601 -> $001; $600 ->$000; $620 ->$020).
;-----

_sPInitRec
    DC.L      _EndsPinitRec - _sPInitRec      ; Block Size
    DC.B      sExec2                         ; code
    DC.B      sCPU68020                      ; revision level
    DC.W      0                               ; reserved
    DC.L      _sPInitProc - *                ; offset to code

    WITH      seBlock, spBlock, SPRAMRecord

_sPInitProc
    sRsrcD3      SET      D3
    spBlockA1     SET      A1
    SEBlockA3     SET      A3
    BaseA2       SET      A2

; MOVE.L      D4, -(SP)
; MOVE.L      #$56780000, D4
; MARK_PROGRESS
; MOVE.L      (SP)+, D4
; MOVE.L      D4, -(SP)
; MOVE.L      #$56560000, D4
; STORE_ERROR
; MOVE.L      (SP)+, D4

    MOVEM.L     A0-A4/D0-D6, -(SP)

    MOVE.W      #1, seStatus(A0)           ; successful initialization
    MOVE.L      A0, SEBlockA3            ; save param block (A0 is destroyed)

; -----
; Turn the slot number into a 32-bit base address.
; -----
    MOVEQ      #0, D0      ; D0 <- 00000000
    MOVE.B      seSlot(SEBlockA3), D0   ; D0 <- 0000000s
    OR.W      #SF0, D0      ; D0 <- 000000Fs
    SWAP      D0      ; D0 <- 00Fs0000
    LSL.L      #8, D0      ; D0 <- Fs000000
    MOVE.L      D0, BaseA2      ; A2 <- Base address to the slot.

; -----
; Reset the hardware -- Set mode to one bit per pixel and Disable video.
; -----
    MOVE.B      #true32b, D0
    _SwapMMUMode          ; Set to 32-bit mode

```

```

; ----- ; 2/26/91 t.s. Added
; Cable Test:
;   1) Turn on video.
;   2) Fill frame buffer with "C0".
;   3) Write Cable Detect regs.
;   4) Check External reg in loop. Loop until composite video bits goes high.
;           C1 - No Monitor Cable
;           C9 - Has Monitor Cable
; -----
;   1) Turn on video and Write Cable Detect regs.
;           Init Maverick Mode A Registers
; ### Make this into a table INIT_MAVERRICK InitMaverick
; llsm 12/6/91 Changed ALL of these Maverick numbers to match the 6x4 monitor.
; llsm 12/6/91 Turn the video on LAST

; Init Maverick Mode A Registers (ts 12/3/90)
MOVE.L    BaseA2, A0          ; get slot base
ADD.L     #kMavModeAReg, A0
MOVE.L    #$COE00000, kMavAHFPorch(A0) ; $03F
MOVE.L    #$EOE00000, kMavAHSync(A0)  ; $01F
MOVE.L    #$BDE00000, kMavAHBPorch(A0); $042 ($043) llsm 12/18/91
MOVE.L    #$32E00000, kMavAHLine(A0) ; $0CD ($0CE) llsm 12/18/91
MOVE.L    #$FDE00000, kMavAVFPorch(A0); $002
MOVE.L    #$FDE00000, kMavAVSync(A0)  ; $002
MOVE.L    #$D9E00000, kMavAVBPorch(A0); $026
MOVE.L    #$20C00000, kMavAFrameLines(A0); $1DF
MOVE.L    #$8DC00000, kMavAHSync22(A0); $172
MOVE.L    #$FFE00000, kMavAColRow(A0) ; $000
MOVE.L    #$15E00000, kMavARowMSBs(A0); $0EA
MOVE.L    #$FF400000, kMavAModesA(A0) ; $500 ($580) llsm 1/14/92
MOVE.L    #$FEE00000, kMavAFS1(A0)   ; $001
MOVE.L    #$F4E00000, kMavAFS2(A0)   ; $00B ($00A) llsm 12/18/91
MOVE.L    #$FFE00000, kMavAReqToTransDly(A0); $000
MOVE.L    #$FFE00000, kMavATransToReqDly(A0); $000

; Init Maverick Mode B Registers (ts 12/3/90)
MOVE.L    BaseA2, A0          ; get slot base
ADD.L     #kMavModeBReg, A0
MOVE.L    #$FCE00000, kMavBExtMode(A0) ; $003 1-bit 6x4
MOVE.L    #$FBEO0000, kMavBRefreshCnt(A0) ; $004
MOVE.L    #$FFE00000, kMavBModesB(A0)  ; $000 llsm 1/16/92
MOVE.L    #$FFE00000, kMavBZoom(A0)   ; $000
MOVE.L    #$EFE00000, kMavBNS(A0)    ; $010
MOVE.L    #$FFE00000, kMavBInterruptClear(A0); $000
MOVE.L    #k30Clock6x4, kMavBExternal(A0); llsm 11/27/91

; -----
; MOVE.L    #$FEE00000, kMavBModesB(A0) ; $001 llsm 1/16/92
;
; _SwapMMUMode ; back to original mode llsm 12/6/91
; CABLE_DETECT ; D0 = 0 if cable llsm 12/6/91
MOVE.L    D0, D2
;
MOVE.B    #true32b, D0      ; llsm 12/6/91
; Set to 32-bit mode llsm 12/6/91
MOVE.L    D0, D6      ; llsm 12/6/91
;
CMP.L    #0, D2      ; llsm 12/6/91
BEQ.S    @IsExternalVideo ; llsm 12/6/91
; Otherwise, is either Bartlett or no monitor found.

; -----
; No Cable
; -----
MOVE.L    kMavBExternal(A0), D3
ANDI.L    #kExtRd_Bartlett_Bit, D3
BEQ.W    @Bartlett
;
MOVE.W    #$0, sRsrcD3      ; No monitor valid.
;
MOVE.L    D6, D0      ; llsm 12/6/91
; back to original mode llsm 12/6/91
_BRA.W    @pErrExit      ; leave with error llsm 12/6/91 added

; -----
; @IsExternalVideo
; -----
; The following logic set D3 to the proper monitor type.
; ts 1/30/91 Added
; -----
CLR.L    D1
MOVEQ    #spBlockSize+8, D0

```

```

        _NewPtr      ,SYS, CLEAR           ;spBlockPtr
        MOVEA.L     A0, spBlockA1

        MOVE.B      seSlot(SEBlockA3), spSlot(spBlockA1)
        MOVE.L      #sizeSPRAMRec, D0
        _NewPtr      ,SYS, CLEAR           ;Allocate an SPRAMRec
        MOVE.L      A0, spResult(spBlockA1)

        MOVE.L      spBlockA1, A0
        _sReadPRAMRec

;$$$$ llsm 12/12/91 Fix sIZER.
; To fix sIZER error, saved vendorUse4 (overwritten by sIZER) in vendorUse6 (unused).
; If 4 and 6 don't match, copy 4 to 3 and 6' to 4.
        MOVEM.L    A0-A1/D0-D1, -(SP)
        CLR.L      D0
        CLR.L      D1
        MOVE.L      spResult(spBlockA1), A0
        MOVE.B      yVendorUse4(A0), D0
        MOVE.B      yVendorUse6(A0), D1
        CMP.B      D0, D1
        BEQ.S      @stseven
        MOVE.B      D0, yVendorUse3(A0)      ; copy 4 to 3
        MOVE.B      D1, yVendorUse4(A0)      ; copy 6 to 4
        MOVE.L      A0, spsPointer(spBlockA1)
        MOVE.L      spBlockA1, A0
        _sPutPRAMRec
        _sReadPRAMRec

@stseven    MOVEM.L    (SP)+, A0-A1/D0-D1
;$$$$ llsm 12/12/91 Fix sIZER.

        MOVE.L      spResult(spBlockA1), A0
        MOVE.B      yVendorUse2(A0), D5      ;Virtual Mode
        MOVE.B      yVendorUse3(A0), sRsrcD3  ;Phys. Monitor
        _DisposPtr

        MOVE.L      spBlockA1, A0
        _DisposPtr

        CMPI.B      #sRsrcTwelve, sRsrcD3   ;rb NOTE 40
        BLE.S      @nothere
        MOVE.B      #0, D5

@nothere    CMPI.B      #sRsrc6x4Video, sRsrcD3
        BEQ.S      @SixByFour
        CMPI.B      #sRsrcVGA, sRsrcD3
        BEQ.W      @SixByFourVGA
        CMPI.B      #sRsrcPortrait, sRsrcD3
        BEQ.W      @Portrait
        CMPI.B      #sRsrcSixty, sRsrcD3    ;rb NOTE 6
        BEQ.W      @EightTwo
        CMPI.B      #sRsrcFifty, sRsrcD3   ;rb NOTE 29
        BEQ.W      @EightOne
        CMPI.B      #sRsrcTwelve, sRsrcD3  ;rb NOTE 29
        BEQ.W      @Twelve
        CMPI.B      #sRsrcSeventy, sRsrcD3 ;rb NOTE 29
        BEQ.W      @EightThree
        CMPI.B      #sRsrc832x624, sRsrcD3
        BEQ.W      @EightStuff
        CMPI.B      #sRsrc1x7, sRsrcD3    ;rb NOTE 29
        BEQ.W      @OneXSeven

; Doesn't match anything, so force it to something.
        MOVE.B      #sRsrc6x4Video, sRsrcD3
; -----
; ----- Init Maverick to 6x4 monitor
; -----
@SixByFour
;       MOVE.B      #sRsrc6x4Video, sRsrcD3      ;D3 to point to 24-bit 6x4 resource

; Init Maverick Mode A Registers (ts 12/3/90)
        MOVE.L      BaseA2, A0                ; get slot base
        ADD.L      #kMavModeAReg, A0
        MOVE.L      #$C0E00000, kMavAHFPorch(A0) ; $03F
        MOVE.L      #$E0E00000, kMavAHSync(A0)  ; $01F
        MOVE.L      #$BDE00000, kMavAHBPorch(A0) ; $042 ($043) llsm 12/18/91
        MOVE.L      #$32E00000, kMavAHLine(A0)   ; $0CD ($0CE) llsm 12/18/91
        MOVE.L      #$FDE00000, kMavAVFPorch(A0) ; $002
        MOVE.L      #$FDE00000, kMavAVSync(A0)  ; $002
        MOVE.L      #$D9E00000, kMavAVBPorch(A0) ; $026
        MOVE.L      #$20C00000, kMavAFramelines(A0); $1DF
        MOVE.L      #$8DC00000, kMavAHSync22(A0) ; $172
        MOVE.L      #$FFE00000, kMavAColRow(A0)  ; $000
        MOVE.L      #$15E00000, kMavARowMSBs(A0); $0EA
        MOVE.L      #$FF400000, kMavAModesA(A0)  ; $500 ($580) llsm 1/14/92
        MOVE.L      #$FEE00000, kMavAFS1(A0)    ; $001

```

```

        MOVE.L      #$F4E00000, kMavAFS2(A0)           ; $00B ($00A) llsm 12/18/91
        MOVE.L      #$FFE00000, kMavAReqToTransDly(A0)    ; $000
        MOVE.L      #$FFE00000, kMavATransToReqDly(A0)    ; $000

; Init Maverick Mode B Registers (ts 12/3/90)
        MOVE.L      BaseA2, A0                      ; get slot base
        ADD.L      #kMavModeBReg, A0
        MOVE.L      #$FCE00000, kMavBExtMode(A0)         ; $003 1-bit 6x4
        MOVE.L      #$FBE00000, kMavBRefreshCnt(A0)       ; $004
        MOVE.L      #$FFE00000, kMavBModesB(A0)          ; $000 llsm 1/16/92
        MOVE.L      #$FFE00000, kMavBZoom(A0)           ; $000
        MOVE.L      #$EEF00000, kMavBNS(A0)             ; $010
        MOVE.L      #$FFE00000, kMavBInterruptClear(A0)   ; $000
        MOVE.L      #k30Clock6x4, kMavBExternal(A0) ; llsm 11/27/91
        BRA.W      @setOneBitMode

; -----
; Init Maverick to 6x4 VGA monitor
; ts 1/24/91 added
; -----
@SixByFourVGA
        MOVE.B      #sRsrcVGA, sRsrcD3            ; D3 to point to 24-bit 6x4 VGA resource

; Init Maverick Mode A Registers
        MOVE.L      BaseA2, A0                      ; get slot base
        ADD.L      #kMavModeAReg, A0
        MOVE.L      #$F0E00000, kMavAHFPorch(A0)        ; $00F
        MOVE.L      #$DOE00000, kMavAHSync(A0)          ; $02F
        MOVE.L      #$EFE00000, kMavAHBPorch(A0)        ; $010
        MOVE.L      #$11E00000, kMavAHLine(A0)          ; $0EE
        MOVE.L      #$FAE00000, kMavAVFPorch(A0)        ; $005
        MOVE.L      #$FEE00000, kMavAVSync(A0)          ; $001
        MOVE.L      #$E1E00000, kMavAVBPorch(A0)        ; $01E
        MOVE.L      #$20C00000, kMavAFrameLines(A0)       ; $1DF
        MOVE.L      #$FC000000, kMavAHSync22(A0)        ; $130
        MOVE.L      #$FFE00000, kMavAColRow(A0)          ; $000
        MOVE.L      #$12E00000, kMavARowMSBs(A0)        ; $0ED
        MOVE.L      #$FF400000, kMavAModesA(A0)          ; $500 ($580) llsm 1/14/92
        MOVE.L      #$FEE00000, kMavAFS1(A0)             ; $001
        MOVE.L      #$F4E00000, kMavAFS2(A0)           ; $00B
        MOVE.L      #$FFE00000, kMavAReqToTransDly(A0)   ; $000
        MOVE.L      #$FFE00000, kMavATransToReqDly(A0)   ; $000

; Init Maverick Mode B Registers (ts 12/3/90)
        MOVE.L      BaseA2, A0                      ; get slot base
        ADD.L      #kMavModeBReg, A0
        MOVE.L      #$FCE00000, kMavBExtMode(A0)         ; $003 1-bit 6x4VGA
        MOVE.L      #'$FBE00000, kMavBRefreshCnt(A0)       ; $004
        MOVE.L      #$FFE00000, kMavBModesB(A0)          ; $000 llsm 1/16/92
        MOVE.L      #$FFE00000, kMavBZoom(A0)           ; $000
        MOVE.L      #$EEF00000, kMavBNS(A0)             ; $010
        MOVE.L      #$FFE00000, kMavBInterruptClear(A0)   ; $000
        MOVE.L      #k25Clock6x4VGA, kMavBExternal(A0) ; llsm 12/18/91
        BRA.W      @setOneBitMode

; -----
; Init Maverick to Portrait monitor
; ts 1/24/91 added
; -----
@Portrait
        MOVE.B      #sRsrcPortrait, sRsrcD3          ; D3 to point to 24-bit Portrait resource

; Init Maverick Mode A Registers
        MOVE.L      BaseA2, A0                      ; get slot base
        ADD.L      #kMavModeAReg, A0
        MOVE.L      #$F0E00000, kMavAHFPorch(A0)        ; $00F
        MOVE.L      #$ECE00000, kMavAHSync(A0)          ; $013
        MOVE.L      #$F0E00000, kMavAHBPorch(A0)        ; $00F ($010)
        MOVE.L      #$92E00000, kMavAHLine(A0)          ; $06D ($06C)
        MOVE.L      #$FDE00000, kMavAVFPorch(A0)        ; $002
        MOVE.L      #$FDE00000, kMavAVSync(A0)          ; $002
        MOVE.L      #$D6E00000, kMavAVBPorch(A0)        ; $029 ($023)
        MOVE.L      #$99800000, kMavAFrameLines(A0)       ; $366
        MOVE.L      #$FEE00000, kMavAHSync22(A0)        ; $001
        MOVE.L      #$FDE00000, kMavAColRow(A0)          ; $000
        MOVE.L      #$16C00000, kMavARowMSBs(A0)        ; $1E9 ($1EC)
        MOVE.L      #$D6600000, kMavAModesA(A0)          ; $429 ($4A9) llsm 1/14/92
        MOVE.L      #$FEE00000, kMavAFS1(A0)             ; $001 ($004) llsm 12/18/91
        MOVE.L      #$F9E00000, kMavAFS2(A0)           ; $006 ($005)
        MOVE.L      #$7FE00000, kMavAReqToTransDly(A0)   ; $080
        MOVE.L      #$7FE00000, kMavATransToReqDly(A0)   ; $080

; Init Maverick Mode B Registers
        MOVE.L      BaseA2, A0                      ; get slot base
        ADD.L      #kMavModeBReg, A0
        MOVE.L      #$F4E00000, kMavBExtMode(A0)         ; $00B 1-bit Portrait
        MOVE.L      #$FAE00000, kMavBRefreshCnt(A0)       ; $005
        MOVE.L      #$FFE00000, kMavBModesB(A0)          ; $000 llsm 1/16/92

```

```

MOVE.L      #$FFE00000, kMavBZoom(A0)          ; $000
MOVE.L      #$_EFE00000, kMavBNS(A0)          ; $010
MOVE.L      #$_FFE00000, kMavBIInterruptClear(A0); $000
MOVE.L      #$_k57ClockPortrait, kMavBExternal(A0); llsm 11/27/91
BRA.W      @setOneBitMode

; -----
; rb NOTE 39   Init Maverick to 12inch monitor   for svga
; -----
@twelve
; MOVE.B      #sRsrcTwelve, sRsrcD3           ; D3 to point to 24-bit 8x6 resource

; Init Maverick Mode A Registers
MOVE.L      BaseA2, A0                      ; get slot base
ADD.L      #kMavModeAReg, A0                ; rb NOTE 9
MOVE.L      #$_EFE00000, kMavAHFPorch(A0)    ; $010
MOVE.L      #$_E7E00000, kMavAHSync(A0)       ; $018
MOVE.L      #$_D5E00000, kMavAHBPorch(A0)     ; $2a
MOVE.L      #$_43E00000, kMavAHLIne(A0)        ; $BC
MOVE.L      #$_FDE00000, kMavAVFPorch(A0)      ; $2
MOVE.L      #$_FDE00000, kMavAVSync(A0)        ; $2
MOVE.L      #$_EDE00000, kMavAVBPorch(A0)       ; $12
MOVE.L      #$_80C00000, kMavAFrameLines(A0)    ; $17F
MOVE.L      #$_FFE00000, kMavAHSync22(A0)      ; $0
MOVE.L      #$_FFE00000, kMavAColRow(A0)        ; $000
MOVE.L      #$_0BE00000, kMavARowMSBs(A0)       ; $F4
MOVE.L      #$_FF400000, kMavAModesA(A0)        ; $500
MOVE.L      #$_FDE00000, kMavAFS1(A0)          ; $002
MOVE.L      #$_FBE00000, kMavAFS2(A0)          ; $004
MOVE.L      #$_FFE00000, kMavAReqToTransDly(A0); $00
MOVE.L      #$_FFE00000, kMavATransToReqDly(A0); $00

; Init Maverick Mode B Registers
MOVE.L      BaseA2, A0                      ; get slot base
ADD.L      #kMavModeBReg, A0                ; rb NOTE 9
MOVE.L      #$_FEE00000, kMavBExtMode(A0)      ; $001
MOVE.L      #$_EBE00000, kMavBRefreshCnt(A0)    ; $014
MOVE.L      #$_FFE00000, kMavBModesB(A0)        ; $000
MOVE.L      #$_FFE00000, kMavBZoom(A0)          ; $000
MOVE.L      #$_FFE00000, kMavBNS(A0)          ; $010
MOVE.L      #$_FFE00000, kMavBIInterruptClear(A0); $000
MOVE.L      #$_kTwelveClock, kMavBExternal(A0); 1
BRA.W      @setOneBitMode

; -----
; rb NOTE 30   Init Maverick to 8x6@56HZ monitor   for svga
; -----
@eightone
; MOVE.B      #sRsrc8x6V56, sRsrcD3           ; D3 to point to 24-bit 8x6 resource

; Init Maverick Mode A Registers
MOVE.L      BaseA2, A0                      ; get slot base
ADD.L      #kMavModeAReg, A0                ; rb NOTE 9
MOVE.L      #$_E7E00000, kMavAHFPorch(A0)    ; $018
MOVE.L      #$_DBE00000, kMavAHSync(A0)       ; $024
MOVE.L      #$_7DE00000, kMavAHBPorch(A0)     ; $82
MOVE.L      #$_E7C00000, kMavAHLIne(A0)        ; $118
MOVE.L      #$_FDE00000, kMavAVFPorch(A0)      ; $2
MOVE.L      #$_FDE00000, kMavAVSync(A0)        ; $2
MOVE.L      #$_F5E00000, kMavAVBPorch(A0)       ; $A
MOVE.L      #$_A9A00000, kMavAFrameLines(A0)    ; $256
MOVE.L      #$_FFE00000, kMavAHSync22(A0)      ; $000
MOVE.L      #$_FFE00000, kMavAColRow(A0)        ; $000
MOVE.L      #$_06C00000, kMavARowMSBs(A0)       ; $1F9
MOVE.L      #$_D6600000, kMavAModesA(A0)        ; $429
MOVE.L      #$_FDE00000, kMavAFS1(A0)          ; $002
MOVE.L      #$_FDE00000, kMavAFS2(A0)          ; $002
MOVE.L      #$_FFE00000, kMavAReqToTransDly(A0); $000
MOVE.L      #$_FFE00000, kMavATransToReqDly(A0); $000

; Init Maverick Mode B Registers
MOVE.L      BaseA2, A0                      ; get slot base
ADD.L      #kMavModeBReg, A0                ; rb NOTE 9
MOVE.L      #$_FEE00000, kMavBExtMode(A0)      ; $001
MOVE.L      #$_EBE00000, kMavBRefreshCnt(A0)    ; $014
MOVE.L      #$_FFE00000, kMavBModesB(A0)        ; $000
MOVE.L      #$_FFE00000, kMavBZoom(A0)          ; $000
MOVE.L      #$_FDE00000, kMavBNS(A0)          ; $002
MOVE.L      #$_FFE00000, kMavBIInterruptClear(A0); $000
MOVE.L      #$_kfiftyClock, kMavBExternal(A0); 
BRA.W      @setOneBitMode

; -----
; rb NOTE 30   Init Maverick to 8x6@60Hz monitor   for svga
; -----
@eighttwo

```

```

; Init Maverick Mode A Registers
MOVE.L      BaseA2, A0          ; get slot base
ADD.L      #kMavModeAReg, A0    ;rb NOTE 9
MOVE.L      #$EDE00000, kMavAHFPorch(A0) ; $012
MOVE.L      #$EBE00000, kMavAHSync(A0)  ; $014
MOVE.L      #$D8E00000, kMavAHBPorch(A0) ; $27
MOVE.L      #$80E00000, kMavAHLLine(A0)   ; $7F
MOVE.L      #$FDE00000, kMavAVFPorch(A0) ; $2
MOVE.L      #$FDE00000, kMavAVSync(A0)   ; $2
MOVE.L      #$FE000000, kMavAVBPorch(A0) ; $10
MOVE.L      #$A9A00000, kMavAFrameLines(A0); $256
MOVE.L      #$FFE00000, kMavAHSync22(A0) ; $000
MOVE.L      #$FFE00000, kMavAColRow(A0)   ; $000
MOVE.L      #$09C00000, kMavARowMSBs(A0); $1F6
MOVE.L      #$D6600000, kMavAModesA(A0)  ; $429
MOVE.L      #$F7E00000, kMavAFS1(A0)     ; $008
MOVE.L      #$F5E00000, kMavAFS2(A0)     ; $00A
MOVE.L      #$7FE00000, kMavAReqToTransDly(A0); $080
MOVE.L      #$7FE00000, kMavATransToReqDly(A0); $080

; Init Maverick Mode B Registers
MOVE.L      BaseA2, A0          ; get slot base
ADD.L      #kMavModeBReg, A0    ;rb NOTE 9
MOVE.L      #$F7E00000, kMavBExtMode(A0) ; $008
MOVE.L      #$EBE00000, kMavBRefreshCnt(A0); $014
MOVE.L      #$FFE00000, kMavBModesB(A0)  ; $000
MOVE.L      #$FFE00000, kMavBZoom(A0)   ; $000
MOVE.L      #$FE000000, kMavBNS(A0)    ; $010
MOVE.L      #$FFE00000, kMavBInterruptClear(A0); $000
MOVE.L      #ksixtyClock, kMavBExternal(A0); F
BRA.W      @setOneBitMode

; -----
; -----
;rb NOTE 30  Init Maverick to 8x6@72Hz monitor  for svga
; -----
@eightthree

; Init Maverick Mode A Registers
MOVE.L      BaseA2, A0          ; get slot base
ADD.L      #kMavModeAReg, A0    ;rb NOTE 9
MOVE.L      #$E0E00000, kMavAHFPorch(A0) ; $01F
MOVE.L      #$DBE00000, kMavAHSync(A0)  ; $024
MOVE.L      #$B9E00000, kMavAHBPorch(A0) ; $46
MOVE.L      #$A3E00000, kMavAHLLine(A0)   ; $5C
MOVE.L      #$F5E00000, kMavAVFPorch(A0) ; $A
MOVE.L      #$F5E00000, kMavAVSync(A0)   ; $A
MOVE.L      #$E7E00000, kMavAVBPorch(A0) ; $18
MOVE.L      #$A9A00000, kMavAFrameLines(A0); $256
MOVE.L      #$FFE00000, kMavAHSync22(A0) ; $000
MOVE.L      #$FFE00000, kMavAColRow(A0)   ; $000
MOVE.L      #$15C00000, kMavARowMSBs(A0); $1EA
MOVE.L      #$D6600000, kMavAModesA(A0)  ; $429
MOVE.L      #$F9E00000, kMavAFS1(A0)     ; $006
MOVE.L      #$F7E00000, kMavAFS2(A0)     ; $008
MOVE.L      #$8FE00000, kMavAReqToTransDly(A0); $070
MOVE.L      #$8FE00000, kMavATransToReqDly(A0); $070

; Init Maverick Mode B Registers
MOVE.L      BaseA2, A0          ; get slot base
ADD.L      #kMavModeBReg, A0    ;rb NOTE 9
MOVE.L      #$F7E00000, kMavBExtMode(A0) ; $008
MOVE.L      #$EBE00000, kMavBRefreshCnt(A0); $014
MOVE.L      #$FFE00000, kMavBModesB(A0)  ; $000
MOVE.L      #$FFE00000, kMavBZoom(A0)   ; $000
MOVE.L      #$FE000000, kMavBNS(A0)    ; $010
MOVE.L      #$FFE00000, kMavBInterruptClear(A0); $000
MOVE.L      #kseventyClock, kMavBExternal(A0); F
BRA.W      @setOneBitMode

; -----
;rb NOTE 6  Init Maverick to 832x624 monitor
; -----
@eightstuff
; MOVE.B      #sRsrc832x624, sRsrcD3      ; D3 to point to 24-bit 832x624 resource

; Init Maverick Mode A Registers
MOVE.L      BaseA2, A0          ; get slot base
ADD.L      #kMavModeAReg, A0    ;rb NOTE 9
MOVE.L      #$F3E00000, kMavAHFPorch(A0) ; $00C
MOVE.L      #$EDE00000, kMavAHSync(A0)  ; $012
MOVE.L      #$B0E00000, kMavAHBPorch(A0) ; $04F
MOVE.L      #$7FE00000, kMavAHLLine(A0)   ; $080
MOVE.L      #$FBE00000, kMavAVFPorch(A0) ; $004
MOVE.L      #$F7E00000, kMavAVSync(A0)   ; $008
MOVE.L      #$FE000000, kMavAVBPorch(A0) ; $010

```

```

MOVE.L      #$92A00000, kMavAFrameLines(A0) ; $26D
MOVE.L      #$FFE00000, kMavAHSync22(A0)   ; $000
MOVE.L      #$FFE00000, kMavAColRow(A0)    ; $000
MOVE.L      #$0DC00000, kMavARowMSBs(A0)  ; $1f2
MOVE.L      #$D6600000, kMavAModesA(A0)   ; $429
MOVE.L      #$FDE00000, kMavAFS1(A0)     ; $002
MOVE.L      #$FBE00000, kMavAFS2(A0)     ; $004
MOVE.L      #$7FE00000, kMavAReqToTransDly(A0) ; $080
MOVE.L      #$7FE00000, kMavATransToReqDly(A0) ; $080

; Init Maverick Mode B Registers
MOVE.L      BaseA2, A0                      ; get slot base
ADD.L      #kMavModeBReg, A0
MOVE.L      #$F7E00000, kMavBExtMode(A0)      ; $008
MOVE.L      #$F9E00000, kMavBRefreshCnt(A0)    ; $006*
MOVE.L      #$FDE00000, kMavBModesB(A0)      ; $000
MOVE.L      #$FFE00000, kMavBZoom(A0)        ; $000
MOVE.L      #$ED000000, kMavBNS(A0)          ; $012*
MOVE.L      #$FFE00000, kMavBInterruptClear(A0); $000
MOVE.L      #k832x624Clock, kMavBExternal(A0); 
BRA.W      @setOneBitMode

; -----
;rb NOTE 30  Init Maverick to 1024x768 monitor
; -----
@onexseven
; MOVE.B      #sRsrc1x7, sRsrcD3           ; D3 to point to 24-bit 832x624 resource

; Init Maverick Mode A Registers
MOVE.L      BaseA2, A0                      ; get slot base
ADD.L      #kMavModeAReg, A0
MOVE.L      #$E2E00000, kMavAHFPorch(A0)    ; $01D
MOVE.L      #$DFF00000, kMavAHSync(A0)       ; $020
MOVE.L      #$C9E00000, kMavAHBPorch(A0)    ; $036
MOVE.L      #$5CE00000, kMavAHLine(A0)      ; $0A3
MOVE.L      #$FBE00000, kMavAVFPorch(A0)    ; $004
MOVE.L      #$FBE00000, kMavAVSync(A0)      ; $004
MOVE.L      #$ED000000, kMavAVBPorch(A0)    ; $012
MOVE.L      #$01A00000, kMavAFrameLines(A0); $2FE
MOVE.L      #$FFE00000, kMavAHSync22(A0)    ; $000
MOVE.L      #$FFE00000, kMavAColRow(A0)     ; $000
MOVE.L      #$0CC00000, kMavARowMSBs(A0)   ; $1F3
MOVE.L      #$D6600000, kMavAModesA(A0)   ; $429
MOVE.L      #$FDE00000, kMavAFS1(A0)     ; $006
MOVE.L      #$7FE00000, kMavAFS2(A0)     ; $008
MOVE.L      #$7FE00000, kMavAReqToTransDly(A0); $080
MOVE.L      #$7FE00000, kMavATransToReqDly(A0); $080

; Init Maverick Mode B Registers
MOVE.L      BaseA2, A0                      ; get slot base
ADD.L      #kMavModeBReg, A0
MOVE.L      #$F7E00000, kMavBExtMode(A0)      ; $008
MOVE.L      #$EBE00000, kMavBRefreshCnt(A0)    ; $014
MOVE.L      #$FDE00000, kMavBModesB(A0)      ; $000
MOVE.L      #$FFE00000, kMavBZoom(A0)        ; $000
MOVE.L      #$F3E00000, kMavBNS(A0)          ; $00C
MOVE.L      #$FFE00000, kMavBInterruptClear(A0); $000
MOVE.L      #k1x7Clock, kMavBExternal(A0); 
BRA.W      @setOneBitMode

; -----
; Init Maverick to Bartlett monitor (ts 1/16/91)
; -----
@Bartlett
; -----
;Disable slot $E
MOVE.M      A0/D0-D1, -(SP)

CLR.L      D1
MOVEQ    _NewPtr, .SYS, CLEAR                ;spBlockPtr

MOVE.B      #$E, spSlot(A0)
MOVE.B      #0, spExtDev(A0)
MOVE.B      #$01, spID(A0)
_sDeleteSRTRec
MOVE.B      #$E, spSlot(A0)
MOVE.B      #0, spExtDev(A0)
MOVE.B      #$80, spID(A0)
_sDeleteSRTRec
MOVE.B      #$E, spSlot(A0)
MOVE.B      #0, spExtDev(A0)
MOVE.B      #$A0, spID(A0)
_sDeleteSRTRec
MOVE.B      #$E, spSlot(A0)
MOVE.B      #0, spExtDev(A0)
MOVE.B      #$FF, spID(A0)
_sDeleteSRTRec

```

```

` _DisposPtr

; Blacken the Internal video frame buffer

MOVE.L #$FEE08040, A0 ; 7/2/91 ts Base Address for internal video
MOVE.L #$1560, D0 ; 7/2/91 ts The Counter
@Blacken MOVE.L #$FFFFFF, (A0)+ ; 7/2/91 ts
DBF D0, @Blacken ; 7/2/91 ts blacken word
; 7/2/91 ts Done? No, Do it some more.

;Force slot $9 to be Main
MOVE.L #$09830983, -(SP)
MOVE.L SP, A0 ;BufferAddress into A0
MOVE.L #$00020080, D0 ;Length, Clock Address
_WriteXPRam
MOVE.L (SP)+, D0

MOVEM.L (SP)+, A0/D0-D1
; -----
MOVE.B #sRsrcBartlett, sRsrcD3 ;D3 to point to 24-bit Bartlett resource

; Init Maverick Mode A Registers
MOVE.L BaseA2, A0 ; get slot base
ADD.L #kMavModeAReg, A0
MOVE.L #$E800000, kMavAHFPorch(A0) ; $014 12/19/91
MOVE.L #$C800000, kMavAHSync(A0) ; $03F 12/19/91
MOVE.L #$F2E00000, kMavAHBPorch(A0) ; $00D 12/19/91
MOVE.L #$62E00000, kMavAHLIne(A0) ; $09D
MOVE.L #$FBE00000, kMavAVFPorch(A0) ; $004
MOVE.L #$FCE00000, kMavAVSync(A0) ; $003
MOVE.L #$EEE00000, kMavAVBPorch(A0) ; $011
MOVE.L #$ABC00000, kMavAFrameLines(A0) ; $154
MOVE.L #$E5C00000, kMavAHSync22(A0) ; $11A
MOVE.L #$FFE00000, kMavAColRow(A0) ; $000
MOVE.L #$0CE00000, kMavARowMSBs(A0) ; $0F3
MOVE.L #$FA400000, kMavAModesA(A0) ; $505 ($585) llsm 1/14/92
MOVE.L #$FEE00000, kMavAFS1(A0) ; $001
MOVE.L #$F4E00000, kMavAFS2(A0) ; $00B
MOVE.L #$FFE00000, kMavAReqToTransDly(A0) ; $000
MOVE.L #$FFE00000, kMavATransToReqDly(A0) ; $000

; Init Maverick Mode B Registers
MOVE.L BaseA2, A0 ; get slot base
ADD.L #kMavModeBReg, A0
MOVE.L #$F0E00000, kMavBExtMode(A0) ; $00F 1-bit Bartlett
MOVE.L #$F8E00000, kMavBRefreshCnt(A0) ; $007
MOVE.L #$D8E00000, kMavBModesB(A0) ; $020
MOVE.L #$FFE00000, kMavBZoom(A0) ; $000
MOVE.L #$SEFE00000, kMavBNS(A0) ; $010
MOVE.L #$FFE00000, kMavBInterruptClear(A0) ; $000
MOVE.L #k15ClockBartlett, kMavBExternal(A0) ; llsm 11/27/91

@setOneBitMode
MOVE.L BaseA2, A0
ADD.L #kMavClutBaseAddr, A0
MOVE.L #1, kMavClutMaskRegOffs(A0) ; 1-bit mode
; -----
; set the color table to black and white
; -----
MOVE #1, D4 ; go thru loop twice
CLR.L D2 ; first entry is black
CLR.L kMavClutWriteOffs(A0) ; select entry 0
@pinit1 MOVE.L D2, kMavClutDataRegOffs(A0) ; red llsm 12/11/91 was kMavClutReadOffs
MOVE.L D2, kMavClutDataRegOffs(A0) ; green llsm 12/11/91
MOVE.L D2, kMavClutDataRegOffs(A0) ; blue llsm 12/11/91
NOT.L D2 ; change black to white

DBRA D4, @pinit1
; -----
; clear video RAM to a nice gray
; -----
MOVE.L BaseA2, A0
MOVE.L #$AAAAAAA, D4

CMP.B #sRsrcPortrait, sRsrcD3
BEQ.S @zeroShort2
CMP.B #sRsrcfifty, sRsrcD3 ;rb NOTE 11
BEQ.S @zeroShort2
CMP.B #sRsrcsixty, sRsrcD3 ;rb NOTE 11
BEQ.S @zeroShort2
CMP.B #sRsrcseventy, sRsrcD3 ;rb NOTE 11
BEQ.S @zeroShort2
CMP.B #sRsrc1x7, sRsrcD3 ;rb NOTE 37

```

```

        BEQ.S      @zeroShort2
        CMP.B      #sRsrc832x624, sRsrcD3
        BEQ.S      @zeroShort2
        CMP.B      #1, D5          ;1Kx1K virtual video
        BEQ.S      @zeroShort2
        MOVE.W     #kRows, D2       ;D0 is rows-1
        BRA.S      @zero0
@zeroShort2
        MOVE.W     #kShortRows, D2    ;D0 is rows-1

@zero0
        CMP.B      #sRsrcPortrait, sRsrcD3
        BEQ.S      @zeroShort
        CMP.B      #sRsrcfifty, sRsrcD3   ;rb NOTE 11
        BEQ.S      @zeroShort
        CMP.B      #sRsrcsixty, sRsrcD3  ;rb NOTE 11
        BEQ.S      @zeroShort
        CMP.B      #sRsrcseventy, sRsrcD3 ;rb NOTE 11
        BEQ.S      @zeroShort
        CMP.B      #sRsrcclx7, sRsrcD3  ;rb NOTE 11
        BEQ.S      @zeroShort
        CMP.B      #sRsrc832x624, sRsrcD3
        BEQ.S      @zeroShort
        CMP.B      #1, D5          ;1Kx1K virtual video
        BEQ.S      @zeroShort
        MOVE.W     #kRowlongs, D1      ; number of longs in a row
        BRA.S      @zerol

@zeroShort
        MOVE.W     #kShortrowlongs, D1   ; number of longs in a row
@zerol
        MOVE.L     D4, (A0)+
        DBRA      D1, @zerol
        NOT.L     D4
        DBRA      D2, @zero0

MOVE.L     D6, D0      ; llsm 12/6/91
                _SwapMMUMode           ; back to original mode

; -----
; The following logic set D3 to the proper monitor type.
; ts 12/18/90 Added
; -----
        CLR.L      D1
        MOVEQ     #spBlockSize+8, D0
        _NewPtr    ,SYS, CLEAR      ;spBlockPtr
        MOVEA.L   A0, spBlockA1

; Do we do virtual monitors?
        MOVE.B     seSlot(SEBlockA3), spSlot(spBlockA1)
        MOVE.L     #sizeSPRAMRec, D0
        _NewPtr    ,SYS, CLEAR      ;Allocate an SPARAMRec
        MOVE.L     A0, spResult(spBlockA1)

        MOVE.L     spBlockA1, A0
                _sReadPRAMRec

;$$$$ llsm 12/12/91 Fix sIZER.
; To fix siZER error. saved vendorUse4 (overwritten by siZER) in vendorUse6 (unused).
; If 4 and 6 don't match, copy 4 to 3 and 6 to 4.
        MOVEM.L   A0-A1/D0-D1, -(SP)
        CLR.L      D0
        CLR.L      D1
        MOVE.L     spResult(spBlockA1), A0
        MOVE.B     yVendorUse4(A0), D0
        MOVE.B     yVendorUse6(A0), D1
        CMP.B      D0, D1
        BEQ.S      @NeverMind2
        MOVE.B     D0, yVendorUse3(A0)   ; copy 4 to 3
        MOVE.B     D1, yVendorUse4(A0)   ; copy 6 to 4
        MOVE.L     A0, spsPointer(spBlockA1)
        MOVE.L     spBlockA1, A0
        _sPutPRAMRec
        _sReadPRAMRec

@NeverMind2
        MOVEM.L   (SP)+, A0-A1/D0-D1
;$$$$ llsm 12/12/91 Fix sIZER.

        MOVE.L     spResult(spBlockA1), A0
        CMP.B     yVendorUse4(A0), sRsrcD3
        BEQ.S      @SkipReset2
        MOVE.B     #0, ScrnInval      ; Invalidate the 'scrn' resource.
        MOVE.B     #$80, yVendorUse1(A0) ; Two Color screen depth.

        MOVE.B     yVendorUse5(A0), D0   ;IRE/Gray
        AND.B     #kVend_ColGray_Mask, D0 ; Clear Gray/Color bit
        MOVE.B     D0, yVendorUse5(A0)   ;IRE/Gray

@SkipReset2

```

```

        MOVE.B      sRsrcD3, yVendorUse4(A0) ; Save REAL monitor type
        MOVE.B      sRsrcD3, yVendorUse6(A0);$$$$ this fixes sIZER error
        CMP.L      #$00, sRsrcD3
        BEQ.W      @SkipVirtual      ;0 = Do Nothin

        MOVE.B      yVendorUse3(A0), D0      ;rb NOTE 40
        CMPI.B      #sRsrcTwelve, D0
        BEQ.S      @nextstep
        CMPI.B      #sRsrcTwelve, D0
        BGT.W      @EscapeVirtual

@nextstep      MOVE.B      yVendorUse2(A0), D0      ; Virtual?
        BEQ.W      @SkipVirtual      ;0 = Do Nothing

        CMPI.B      #sRsrc6x4Video, sRsrcD3 ; llsm 12/20/91
        BNE.S      @Not6x4Virtual      ; llsm 12/20/91
        MOVE.L      BaseA2, A4      ; get slot base llsm 12/20/91
        ADD.L      #kMavModeAReg, A4      ; llsm 12/20/91
        MOVE.L      D0, -(SP)      ; llsm 12/20/91
        MOVE.B      #true32b, D0      ; llsm 12/20/91
        _SwapMMUMode      ;Set to 32-bit mode llsm 12/20/91
        MOVE.L      #$FEE00000, kMavAVFPorch(A4) ; $001 llsm 12/20/91 was in SetMode
        _SwapMMUMode      ; back to original mode llsm 12/20/91
        MOVE.L      (SP)+, D0      ; llsm 12/20/91
        _SwapMMUMode      ; llsm 12/20/91

@Not6x4Virtual      CMP.B      #kMonitorVV1kx512, D0
        BEQ.S      @oneKBy512

; Adjust Maverick Registers for 1k x 1k
        MOVE.L      BaseA2, A4      ; get slot base
        ADD.L      #kMavModeAReg, A4

        MOVE.B      #true32b, D0      ;Set to 32-bit mode

        CMPI.B      #sRsrcBartlett, sRsrcD3
        BEQ.S      @vBartlett
        CMPI.B      #sRsrcTwelve, sRsrcD3
        BEQ.S      @vmyapptwe

        MOVE.L      #$0BC00000, kMavARowMSBs(A4) ; $1EB
        MOVE.L      #$D7400000, kMavAModesA(A4)      ; $528 ($5A8) llsm 1/14/92
        BRA.S      @vContinue

@vmyapptwe      MOVE.L      #$EFA00000, kMavARowMSBs(A4) ; $1F4
        MOVE.L      #$FFA00000, kMavAColRow(A4)      ; $200
        MOVE.L      #$D7400000, kMavAModesA(A4)      ; $528 ($5A8) llsm 1/14/92
        BRA.S      @vContinue

@vBartlett      MOVE.L      #$FFA00000, kMavAColRow(A4)      ; $200
        MOVE.L      #$0CC00000, kMavARowMSBs(A4) ; $1F3
        MOVE.L      #$D2400000, kMavAModesA(A4)      ; $52D ($5AD) llsm 1/14/92

@vContinue      _SwapMMUMode      ; back to original mode

        MOVE.B      #sRsrcVV1kx1k, sRsrcD3 ;1 = 1k by 1k virtual
        BRA.S      @SkipVirtual      ;Note: 2 = 2k by 1k. Is not available on this card.

@oneKBy512      CMP.B      #sRsrcPortrait, sRsrcD3
        BEQ.S      @EscapeVirtual
        MOVE.B      #sRsrcVV1kx512, sRsrcD3 ;3 = 1k by 512 virtual
        BRA.S      @SkipVirtual

@EscapeVirtual      MOVE.B      #0, yVendorUse2(A0)

@SkipVirtual      MOVE.L      A0, spsPointer(spBlockA1)      ;Write out REAL monitor type
        MOVE.L      spBlockA1, A0
        _SPutPRAMRec      MOVE.L      spsPointer(spBlockA1), A0
        _DisposePtr

; -----
; The following logic deletes the "other" video resources.
; ts 12/14/90 Added
; -----
        _sVersion
        BNE.S      @noQD32
;32 bit quick draw is in ROM
;at this point, D3 should contain sRsrc6x4Video32
        ADDI.B      #k32QDOffset, sRsrcD3 ;D3 to point to 32-bit resource

@noQD32 ;no 32 bit Quick Draw in ROM
;at this point, D3 should contain the proper resource number.
        MOVE.B      #kFirstMonitor, D2
        MOVE.L      spBlockA1, A0 ; llsm added 11/27/91

```

```

@bye
    CMP.B      D2, sRsrcD3
    BEQ.S      @addone
    MOVE.B      seSlot(SEBlockA3), spSlot(spBlockA1) ; llsm A1 was A0
    MOVE.B      #0, spExtDev(spBlockA1)
    MOVE.B      D2, spID(spBlockA1)
    _DeleteSRTRec
@addone
    ADDI.B     #1, D2
    CMPI.B     #152, D2           ;rb NOTE 7
    BLE.S      @bye             ;rb NOTE 23 NOTE 31
    MOVE.W     #1, D1

    BRA.S      @exit            ; llsm 12/6/91 added
@pErrExit
    MOVE.W     #kError, D1        ; D1 = init fail code llsm 12/6/91 added
    MOVE.W     D1, seStatus(SEBlockA3) ; llsm 12/6/91
    MOVEM.L    (SP)+, A0-A4/D0-D6 ; llsm 12/6/91
    RTS         ; llsm 12/6/91

@exit
    MOVE.W     D1, seStatus(SEBlockA3)
    MOVEA.L    spBlockA1, A0
    _DispsoPtr

    MOVEM.L    (SP)+, A0-A4/D0-D6
    RTS

_EndsPinitRec EQU      *

;-----
;SecondaryInit
;DESCRIPTION:
;  
SecondaryInit _sSInitRec is an sExecBlock (C&D 8-3). ~  
The code below attempts to implement the secondary init code as spelled ~  
out in Apple Computer's document "32-Bit Video Drivers - Rev 3.1".  
DECLARATION:  
_sSInitProc;  
ARGUMENTS:  
On Entry      A0      Ptr to seBlock.  
On Exit       seStatus field = +1 if successful init.  
              seStatus field = -1 if error during init.  
EXAMPLE:  
N/A.  
HISTORY:  
121490  ts   Created.  
121790  ts   Add 32 byte offset to base video address.  
-----
_sSInitRec DC.L      _EndsSInitRec - _sSInitRec ;Block Size
DC.B      sExec2          ;code
DC.B      sCPU68020        ;revision level
DC.W      0                ;reserved
DC.L      _sSInitProc - *  ;offset to code

_sSInitProc
sRsrcD3      SET      D3
spBlockA1     SET      A1
SEBlockA2     SET      A2

    MOVEM.L    A0-A3/D0-D4, -(SP)
    MOVE.L     A0, SEBlockA2        ;A2 Ptr to seBlock

    ; build devBaseAddr from slot
    MOVE.B     seSlot(SEBlockA2), D4
    EXTB.L    D4
    ROR.L     #8, D4
    ORI.L     #$F0000000, D4      ;Fs00 0000 for 32-bit mode

    MOVEQ     #spBlockSize+8, D0
    _NewPtr    ,SYS, CLEAR
    MOVEA.L    A0, spBlockA1        ;A1 Ptr to spBlock
    ; determine if 32-bit Quickdraw is in ROM
    _sVersion   ;returns smSelOOBErr =-338, if no QD32
    BNE      @goodexit
    CMPI.L    #0, spResult(spBlockA1)
    BGT.S     @sMgrOK            ;positive spResult=qd32 here
    BRA.W     @sErrExit          ; llsm 12/6/91 added
@sMgrOK
    CMPI.L    #2, spResult(spBlockA1)
    BEQ      @goodexit          ;32QD in ROM exit

@RAMSMgr
    MOVE.W     #$AB03, D0          ;32QD trap Address
    _GetTrapAddress
    MOVE.L     A0, D1

```

```

        CMPI.L      #kDnimplemented, D1
        BEQ         @goodexit

; Do we do virtual monitors?
        MOVE.B      seSlot(SEBlockA2), spSlot(spBlockA1)
        MOVE.L      #sizeSPRAMRec, D0
        _NewPtr     ,SYS, CLEAR          ;Allocate an SPRAMRec
        MOVE.L      A0, spResult(spBlockA1)

        MOVE.L      spBlockA1, A0
        _sReadPRAMRec
        MOVE.L      spResult(spBlockA1), A0

        MOVE.B      yVendorUse4(A0), sRsrcD3    ;Get Actual monitor resource

        CMP.B      #sRsrcBartlett, sRsrcD3   ; Is this Bartlett?
        BNE.S      @SkipBartlett           ; If No, skip.

; kill the internal video
        MOVE.L      SInfoPtr, A3           ; Pointer to Slot info table
        MOVE.L      $D*4(A3), $E*4(A3)     ; Wipe out entry for Slot $E

        MOVE.L      #$FEE08040, A3         ; 7/2/91 ts Base Address for internal video
        MOVE.L      #$1560, D0             ; 7/2/91 ts The Counter
@Blacken      MOVE.L      #$FFFFFFFFF, (A3)+       ; 7/2/91 ts blacken word
        DBF         D0, @Blacken          ; 7/2/91 ts Done? No, Do it some more.

@SkipBartlett
        MOVE.B      yVendorUse2(A0), D0      ;Check for virtual monitor
        BEQ.S      @FinishVirtual         ;0 = Do Nothing
        CMP.B      #kMonitorVV1kx512, D0
        BEQ.S      @OneKBy512
        MOVE.B      #sRsrcVV1kx1k, sRsrcD3 ;1 = 1k by 1k virtual
                                         ;Note: 2 = 2k by 1k. Is not available on this card.

        BRA.S      @FinishVirtual

@OneKBy512
        MOVE.B      #sRsrcVV1kx512, sRsrcD3 ;3 = 1k by 512 virtual

@FinishVirtual
        _DisposPtr   ;Get rid of SPRAMRecord

        MOVEA.L     spBlockA1, A0          ;put spslot pointer in A0
        MOVE.B      seSlot(SEBlockA2), spSlot(spBlockA1)
        MOVE.B      #0, spExtDev(spBlockA1)
        MOVE.B      sRsrcD3, spID(spBlockA1) ;deletes 24 bit resource

        _sRsrcInfo
        _sDeleteSRTRec
        ADDI.B      #k32QDOffset, sRsrcD3
        MOVE.L      $0, spsPointer(spBlockA1)
        MOVE.B      sRsrcD3, spID(spBlockA1) ;enable 32 bit resource

        MOVE.L      #0, spParamData(spBlockA1)

        _InsertSRTRec
        CMPI.W      #0, spRefNum(spBlockA1) ;0=not startup device
        BEQ.S      @goodexit

        CLR.L      -(SP)
        _GetGDevice
        MOVE.L      (SP)+, A0            ;get the gDevice handle
        BEQ.S      @sErrExit           ; llsm 12/6/91 added
        MOVE.L      (A0), A0
        MOVE.L      gdPMap(A0), A0       ;correct gDevice info if
        MOVE.L      (A0), A0             ;start-up device
        ADDI.L      #1024, D4           ;Add 32 byte offset to video base - ts 12/17/90
        MOVE.L      D4, pmBaseAddr(A0)  ;D4 IS BASE ADDRESS

@goodExit
; Are we in Bartlett? If so fix the gray stuff.
        MOVE.B      seSlot(SEBlockA2), spSlot(spBlockA1)
        MOVE.L      #sizeSPRAMRec, D0
        _NewPtr     ,SYS, CLEAR          ;Allocate an SPRAMRec
        MOVE.L      A0, spResult(spBlockA1)

        MOVE.L      spBlockA1, A0
        _sReadPRAMRec
        MOVE.L      spResult(spBlockA1), A0

        MOVE.B      yVendorUse4(A0), sRsrcD3    ;Get Actual monitor resource
        _DisposPtr   ;Get rid of SPRAMRecord

        CMP.B      #sRsrcBartlett, sRsrcD3   ; Is this Bartlett?
        BNE.S      @SkipOut               ; If No, skip.

        MOVEM.L     A0/D0-D1, -(SP)
        CLR.L      D0
        MOVE.B      yVendorUse5(A0), D0      ;Get Gray/Color bit
        CLR.L      -(SP)
        _GetGDevice

```

```

        MOVE.L      (SP)+, A0          ;get the gDevice handle
        MOVE.L      (A0), A0
        MOVE.W      gdFlags(A0), D1    ;correct gDevice info if
        LSR.B      #1, D0
        AND.W      #1, D0
        AND.W      #$FFFE, D1
        ADD.W      D0, D1
        MOVE.W      D1, gdFlags(A0)

        MOVEM.L     (SP)+, A0/D0-D1

@SkipOut
        MOVE.W      #1, D1

@sErrExit
        BRA.S      @exit      ; llsm 12/6/91 added
        MOVE.W      #kError, D1    ; llsm 12/6/91 added

@exit
        MOVE.W      D1, seStatus(SEBlockA2)
        MOVE.L      spBlockA1, A0
        _DisposPtr

        MOVEM.L     (SP)+, A0-A3/D0-D4
        RTS

_EndsSinitRec EQU      *
ENDWITH

_VendorInfo OSLstEntry      vendorId, _vendorId
OSLstEntry      revLevel, _revLevel
OSLstEntry      partNum, _partNum
DatLstEntry      endOfList, 0

_vendorId DC.L      'XCEED Technology, Inc', 0
_revLevel myRevLev
_partNum DC.L      '0369', 0

;-----
;sResourceList
;DESCRIPTION:
;    List of sResources -- all devices supported and their depths.
;DECLARATION:
;    N/A.
;ARGUMENTS:
;    N/A.
;EXAMPLE:
;    N/A.
;HISTORY:
;    121490 ts      Add *_sRsrc6x4Video32*
;-----
_sRsrc6x4Video OSLstEntry      sRsrcType, _VideoType
OSLstEntry      sRsrcName, _VideoName

IF &DriverOnChip THEN
    OSLstEntry      sRsrcDrvrDir, _VidDrvrDir
ELSE
    OSLstEntry      sRsrcLoadRec, _dLoadRec
ENDIF

DatLstEntry      sRsrcFlags, kFOpenAtStartBit
DatLstEntry      sRsrcHWDevId, 1           ; Unique hardware device ID (????)
OSLstEntry      MinorBaseOS, _MinorBase
OSLstEntry      MinorLength, _MinorLength
OSLstEntry      OneBitMode, _OneBitMode
OSLstEntry      TwoBitMode, _TwoBitMode
OSLstEntry      FourBitMode, _FourBitMode
OSLstEntry      EightBitMode, _EightBitMode
DatLstEntry      endOfList, 0

_sRsrcVGA OSLstEntry      sRsrcType, _VideoType
OSLstEntry      sRsrcName, _VideoName

IF &DriverOnChip THEN
    OSLstEntry      sRsrcDrvrDir, _VidDrvrDir
ELSE
    OSLstEntry      sRsrcLoadRec, _dLoadRec
ENDIF

DatLstEntry      sRsrcFlags, kFOpenAtStartBit
DatLstEntry      sRsrcHWDevId, 1           ; Unique hardware device ID (????)
OSLstEntry      MinorBaseOS, _MinorBase
OSLstEntry      MinorLength, _MinorLength
OSLstEntry      OneBitMode, _OneVGAMode
OSLstEntry      TwoBitMode, _TwoVGAMode
OSLstEntry      FourBitMode, _FourVGAMode
OSLstEntry      EightBitMode, _EightVGAMode

```

```

        DatLstEntry    endOfList, 0

_sRsrcPortrait OSLstEntry    sRsrcType, _VideoType
                OSLstEntry    sRsrcName, _VideoName

        IF &DriverOnChip THEN
            OSLstEntry    sRsrcDrvrDir, _VidDrvrDir
        ELSE
            OSLstEntry    sRsrcLoadRec, _dLoadRec
        ENDIF

        DatLstEntry    sRsrcFlags, kFOpenAtStartBit
        DatLstEntry    sRsrcHWDevId, 1                      ; Unique hardware device ID (???) 
        OSLstEntry    MinorBaseOS, _MinorBase
        OSLstEntry    MinorLength, _MinorLength
        OSLstEntry    OneBitMode, _OnePortMode
        OSLstEntry    TwoBitMode, _TwoPortMode
        OSLstEntry    FourBitMode, _FourPortMode
        DatLstEntry    endOfList, 0

_sRsrcBartlett OSLstEntry    sRsrcType, _VideoType
                OSLstEntry    sRsrcName, _VideoName

        IF &DriverOnChip THEN
            OSLstEntry    sRsrcDrvrDir, _VidDrvrDir
        ELSE
            OSLstEntry    sRsrcLoadRec, _dLoadRec
        ENDIF

        DatLstEntry    sRsrcFlags, kFOpenAtStartBit
        DatLstEntry    sRsrcHWDevId, 1                      ; Unique hardware device ID (???) 
        OSLstEntry    MinorBaseOS, _MinorBase
        OSLstEntry    MinorLength, _MinorLength
        OSLstEntry    OneBitMode, _OneBartMode
        OSLstEntry    TwoBitMode, _TwoBartMode
        OSLstEntry    FourBitMode, _FourBartMode
        OSLstEntry    EightBitMode, _EightBartMode
        DatLstEntry    endOfList, 0

_sRsrcVV1kx1k   OSLstEntry    sRsrcType, _VideoType
                OSLstEntry    sRsrcName, _VideoName

        IF &DriverOnChip THEN
            OSLstEntry    sRsrcDrvrDir, _VidDrvrDir
        ELSE
            OSLstEntry    sRsrcLoadRec, _dLoadRec
        ENDIF

        DatLstEntry    sRsrcFlags, kFOpenAtStartBit
        DatLstEntry    sRsrcHWDevId, 1                      ; Unique hardware device ID (???) 
        OSLstEntry    MinorBaseOS, _MinorBase
        OSLstEntry    MinorLength, _MinorLength
        OSLstEntry    OneBitMode, _OneV1kx1kMode
        OSLstEntry    TwoBitMode, _TwoV1kx1kMode
        OSLstEntry    FourBitMode, _FourV1kx1kMode
        DatLstEntry    endOfList, 0

_sRsrcVV1kx512  OSLstEntry    sRsrcType, _VideoType
                OSLstEntry    sRsrcName, _VideoName

        IF &DriverOnChip THEN
            OSLstEntry    sRsrcDrvrDir, _VidDrvrDir
        ELSE
            OSLstEntry    sRsrcLoadRec, _dLoadRec
        ENDIF

        DatLstEntry    sRsrcFlags, kFOpenAtStartBit
        DatLstEntry    sRsrcHWDevId, 1                      ; Unique hardware device ID (???) 
        OSLstEntry    MinorBaseOS, _MinorBase
        OSLstEntry    MinorLength, _MinorLength
        OSLstEntry    OneBitMode, _OneV1kx512Mode
        OSLstEntry    TwoBitMode, _TwoV1kx512Mode
        OSLstEntry    FourBitMode, _FourV1kx512Mode
        OSLstEntry    EightBitMode, _EightV1kx512Mode
        DatLstEntry    endOfList, 0

_sRsrctwelve   OSLstEntry    sRsrcType, _VideoType ;rb new table
                OSLstEntry    sRsrcName, _VideoName

        IF &DriverOnChip THEN
            OSLstEntry    sRsrcDrvrDir, _VidDrvrDir
        ELSE
            OSLstEntry    sRsrcLoadRec, _dLoadRec
        ENDIF

        DatLstEntry    sRsrcFlags, kFOpenAtStartBit
        DatLstEntry    sRsrcHWDevId, 1                      ; Unique hardware device ID (???) 
        OSLstEntry    MinorBaseOS, _MinorBase

```

```

        OSLstEntry      MinorLength, _MinorLength
        OSLstEntry      OneBitMode, _OneTweMode
        OSLstEntry      TwoBitMode, _TwoTweMode
        OSLstEntry      FourBitMode, _FourTweMode
        OSLstEntry      EightBitMode, _EightTweMode
        DatLstEntry     endOfList, 0
_sRsrc8x6      OSLstEntry      sRsrcType, _VideoType    ;rb new table
                OSLstEntry      sRsrcName, _VideoName

        IF &DriverOnChip THEN
                OSLstEntry      sRsrcDrvDir, _VidDrvDir
        ELSE
                OSLstEntry      sRsrcLoadRec, _dLoadRec
        ENDIF

                DatLstEntry     sRsrcFlags, kFOpenAtStartBit
                DatLstEntry     sRsrcHWDevId, 1           ; Unique hardware device ID (???) 
                OSLstEntry      MinorBaseOS, _MinorBase
                OSLstEntry      MinorLength, _MinorLength
                OSLstEntry      OneBitMode, _One8x6Mode
                OSLstEntry      TwoBitMode, _Two8x6Mode
                OSLstEntry      FourBitMode, _Four8x6Mode
                DatLstEntry     endOfList, 0

_sRsrc832x64   OSLstEntry      sRsrcType, _VideoType    ;rb new table
                OSLstEntry      sRsrcName, _VideoName

        IF &DriverOnChip THEN
                OSLstEntry      sRsrcDrvDir, _VidDrvDir
        ELSE
                OSLstEntry      sRsrcLoadRec, _dLoadRec
        ENDIF

                DatLstEntry     sRsrcFlags, kFOpenAtStartBit
                DatLstEntry     sRsrcHWDevId, 1           ; Unique hardware device ID (???) 
                OSLstEntry      MinorBaseOS, _MinorBase
                OSLstEntry      MinorLength, _MinorLength
                OSLstEntry      OneBitMode, _One832x624Mode
                OSLstEntry      TwoBitMode, _Two832x624Mode
                OSLstEntry      FourBitMode, _Four832x624Mode
                DatLstEntry     endOfList, 0
_sRsrc1x7      OSLstEntry      sRsrcType, _VideoType    ;rb new table
                OSLstEntry      sRsrcName, _VideoName

        IF &DriverOnChip THEN
                OSLstEntry      sRsrcDrvDir, _VidDrvDir
        ELSE
                OSLstEntry      sRsrcLoadRec, _dLoadRec
        ENDIF

                DatLstEntry     sRsrcFlags, kFOpenAtStartBit
                DatLstEntry     sRsrcHWDevId, 1           ; Unique hardware device ID (???) 
                OSLstEntry      MinorBaseOS, _MinorBase
                OSLstEntry      MinorLength, _MinorLength
                OSLstEntry      OneBitMode, _One1x7Mode
                OSLstEntry      TwoBitMode, _Two1x7Mode
                OSLstEntry      FourBitMode, _Four1x7Mode
                DatLstEntry     endOfList, 0

_sRsrc6x4Video32 OSLstEntry      sRsrcType, _VideoType
                  OSLstEntry      sRsrcName, _VideoName

        IF &DriverOnChip THEN
                OSLstEntry      sRsrcDrvDir, _VidDrvDir
        ELSE
                OSLstEntry      sRsrcLoadRec, _dLoadRec
        ENDIF

                DatLstEntry     sRsrcFlags, kFOpenAtStartBit + kf32BitModeBit
                DatLstEntry     sRsrcHWDevId, 1           ; Unique hardware device ID (???) 
                OSLstEntry      MinorBaseOS, _MinorBase
                OSLstEntry      MinorLength, _MinorLength
                OSLstEntry      OneBitMode, _OneBitMode
                OSLstEntry      TwoBitMode, _TwoBitMode
                OSLstEntry      FourBitMode, _FourBitMode
                OSLstEntry      EightBitMode, _EightBitMode
                DatLstEntry     endOfList, 0

_sRsrcVGA32    OSLstEntry      sRsrcType, _VideoType
                  OSLstEntry      sRsrcName, _VideoName

        IF &DriverOnChip THEN
                OSLstEntry      sRsrcDrvDir, _VidDrvDir
        ELSE
                OSLstEntry      sRsrcLoadRec, _dLoadRec
        ENDIF

```

```

        DatLstEntry    sRsrcFlags, kFOpenAtStartBit + kF32BitModeBit
        DatLstEntry    sRsrcHWDevId, 1 ; Unique hardware device ID (??)
        OSLstEntry    MinorBaseOS, _MinorBase
        OSLstEntry    MinorLength, _MinorLength
        OSLstEntry    OneBitMode, _OneVGAMode
        OSLstEntry    TwoBitMode, _TwoVGAMode
        OSLstEntry    FourBitMode, _FourVGAMode
        OSLstEntry    EightBitMode, _EightVGAMode
        DatLstEntry    endOfList, 0

_sRsrcPortrait32    OSLstEntry    sRsrcType, _VideoType
                    OSLstEntry    sRsrcName, _VideoName

IF &DriverOnChip THEN
    OSLstEntry    sRsrcDrvrDir, _VidDrvrDir
ELSE
    OSLstEntry    sRsrcLoadRec, _dLoadRec
ENDIF

        DatLstEntry    sRsrcFlags, kFOpenAtStartBit + kF32BitModeBit
        DatLstEntry    sRsrcHWDevId, 1 ; Unique hardware device ID (??)
        OSLstEntry    MinorBaseOS, _MinorBase
        OSLstEntry    MinorLength, _MinorLength
        OSLstEntry    OneBitMode, _OnePortMode
        OSLstEntry    TwoBitMode, _TwoPortMode
        OSLstEntry    FourBitMode, _FourPortMode
        DatLstEntry    endOfList, 0

_sRsrcBartlett32    OSLstEntry    sRsrcType, _VideoType
                    OSLstEntry    sRsrcName, _VideoName

IF &DriverOnChip THEN
    OSLstEntry    sRsrcDrvrDir, _VidDrvrDir
ELSE
    OSLstEntry    sRsrcLoadRec, _dLoadRec
ENDIF

        DatLstEntry    sRsrcFlags, kFOpenAtStartBit + kF32BitModeBit
        DatLstEntry    sRsrcHWDevId, 1 ; Unique hardware device ID (??)
        OSLstEntry    MinorBaseOS, _MinorBase
        OSLstEntry    MinorLength, _MinorLength
        OSLstEntry    OneBitMode, _OneBartMode
        OSLstEntry    TwoBitMode, _TwoBartMode
        OSLstEntry    FourBitMode, _FourBartMode
        OSLstEntry    EightBitMode, _EightBartMode
        DatLstEntry    endOfList, 0

_sRsrcVV1kx1k32 OSLstEntry .    sRsrcType, _VideoType
                    OSLstEntry    sRsrcName, _VideoName

IF &DriverOnChip THEN
    OSLstEntry    sRsrcDrvrDir, _VidDrvrDir
ELSE
    OSLstEntry    sRsrcLoadRec, _dLoadRec
ENDIF

        DatLstEntry    sRsrcFlags, kFOpenAtStartBit + kF32BitModeBit
        DatLstEntry    sRsrcHWDevId, 1 ; Unique hardware device ID (??)
        OSLstEntry    MinorBaseOS, _MinorBase
        OSLstEntry    MinorLength, _MinorLength
        OSLstEntry    OneBitMode, _OneV1kx1kMode
        OSLstEntry    TwoBitMode, _TwoV1kx1kMode
        OSLstEntry    FourBitMode, _FourV1kx1kMode
        DatLstEntry    endOfList, 0

_sRsrcVV1kx51232    OSLstEntry    sRsrcType, _VideoType
                    OSLstEntry    sRsrcName, _VideoName

IF &DriverOnChip THEN
    OSLstEntry    sRsrcDrvrDir, _VidDrvrDir
ELSE
    OSLstEntry    sRsrcLoadRec, _dLoadRec
ENDIF

        DatLstEntry    sRsrcFlags, kFOpenAtStartBit + kF32BitModeBit
        DatLstEntry    sRsrcHWDevId, 1 ; Unique hardware device ID (??)
        OSLstEntry    MinorBaseOS, _MinorBase
        OSLstEntry    MinorLength, _MinorLength
        OSLstEntry    OneBitMode, _OneV1kx512Mode
        OSLstEntry    TwoBitMode, _TwoV1kx512Mode
        OSLstEntry    FourBitMode, _FourV1kx512Mode
        OSLstEntry    EightBitMode, _EightV1kx512Mode
        DatLstEntry    endOfList, 0

_sRsrctwelveQ32 OSLstEntry    sRsrcType, _VideoType ;rb new table
                    OSLstEntry    sRsrcName, _VideoName

IF &DriverOnChip THEN
    OSLstEntry    sRsrcDrvrDir, _VidDrvrDir

```

```

ELSE
  OSLstEntry      sRsrcLoadRec, _dLoadRec
ENDIF

DatLstEntry      sRsrcFlags, kFOpenAtStartBit + kF32BitModeBit
DatLstEntry      sRsrcHWDevId, 1           ; Unique hardware device ID (??)
OSLstEntry       MinorBaseOS, _MinorBase
OSLstEntry       MinorLength, _MinorLength
OSLstEntry       OneBitMode, _OneTweMode
OSLstEntry       TwoBitMode, _TwoTweMode
OSLstEntry       FourBitMode, _FourTweMode
OSLstEntry       EightBitMode, _EightTweMode
DatLstEntry      endOfList, 0
_sRsrc8x6Q32    OSLstEntry      sRsrcType, _VideoType   ;rb new table
OSLstEntry      sRsrcName, _VideoName

IF &DriverOnChip THEN
  OSLstEntry      sRsrcDrvDir, _VidDrvDir
ELSE
  OSLstEntry      sRsrcLoadRec, _dLoadRec
ENDIF

DatLstEntry      sRsrcFlags, kFOpenAtStartBit + kF32BitModeBit
DatLstEntry      sRsrcHWDevId, 1           ; Unique hardware device ID (??)
OSLstEntry       MinorBaseOS, _MinorBase
OSLstEntry       MinorLength, _MinorLength
OSLstEntry       OneBitMode, _One8x6Mode
OSLstEntry       TwoBitMode, _Two8x6Mode
OSLstEntry       FourBitMode, _Four8x6Mode
DatLstEntry      endOfList, 0

_sRsrc832x624Q32  OSLstEntry      sRsrcType, _VideoType   ;rb new table
OSLstEntry      sRsrcName, _VideoName

IF &DriverOnChip THEN
  OSLstEntry      sRsrcDrvDir, _VidDrvDir
ELSE
  OSLstEntry      sRsrcLoadRec, _dLoadRec
ENDIF

DatLstEntry      sRsrcFlags, kFOpenAtStartBit + kF32BitModeBit
DatLstEntry      sRsrcHWDevId, 1           ; Unique hardware device ID (??)
OSLstEntry       MinorBaseOS, _MinorBase
OSLstEntry       MinorLength, _MinorLength
OSLstEntry       OneBitMode, _One832x624Mode
OSLstEntry       TwoBitMode, _Two832x624Mode
OSLstEntry       FourBitMode, _Four832x624Mode
DatLstEntry      endOfList, 0

_sRsrc1x7Q32     OSLstEntry      sRsrcType, _VideoType   ;rb new table
OSLstEntry      sRsrcName, _VideoName

IF &DriverOnChip THEN
  OSLstEntry      sRsrcDrvDir, _VidDrvDir
ELSE
  OSLstEntry      sRsrcLoadRec, _dLoadRec
ENDIF

DatLstEntry      sRsrcFlags, kFOpenAtStartBit + kF32BitModeBit
DatLstEntry      sRsrcHWDevId, 1           ; Unique hardware device ID (??)
OSLstEntry       MinorBaseOS, _MinorBase
OSLstEntry       MinorLength, _MinorLength
OSLstEntry       OneBitMode, _One1x7Mode
OSLstEntry       TwoBitMode, _Two1x7Mode
OSLstEntry       FourBitMode, _Four1x7Mode
DatLstEntry      endOfList, 0

_VideoType       DC.W          catDisplay, typVideo
DC.W            drSwApple           ;<DrvrSw>
DC.W            kColor8DeviceID    ;<DrvrHw>

_VideoName       DC.L          'XCEED Color 30HR™', 0

IF &DriverOnChip THEN
_VidDrvDir      OSLstEntry      sMacOS68020, _sMacOS68020
DatLstEntry      endOfList, 0

_sMacOS68020     DC.L          _End020Drv->_sMacOS68020
INCLUDE          'theDriver.a'
_Equ             *
ENDIF

INCLUDE          '369Loader.a'

_MinorBase       DC.L          $0000          ; this just seems to work
_MinorLength     DC.L          $12*1024        ; 512K

```

struct TF\_PiNReset st

Steve, Pinname[20]

Steve, meas[2].result

Void printscreenschar B

Byte ConvertToAscii Echo

return

num = ConvertToAscii('A')

```

; -----
; One-bits-per-pixel parameter list
; -----
_OneBitMode OSLstEntry mVidParams, _OneVidParams
DatLstEntry mPageCnt, 1
DatLstEntry mDevType, 0 ; 0 is CLUT Type device
DatLstEntry endOfList, 0

_OneVidParams DC.L _EndOneVParams-_OneVidParams ; Physical block size
DC.L kMyBaseOffset ; base offset for start of screen memory
DC.W kMyRowBytes ; rowbytes
DC.W 0,0,480,640 ; bounds (top,left,bottom,right)
DC.W 0 ; version
DC.W 0 ; packType (indicates to use default)
DC.L 0 ; packSize (reserved for future use)
DC.W 72,0 ; horizontal pixels per inch
DC.W 72,0 ; vertical pixels per inch
DC.W 0 ; pixel type is chunky
DC.W 1 ; PixelSize is 1 bits
DC.W 1 ; 1 component
DC.W 1 ; component size is 1 bit
DC.L 0 ; defPlaneBytes
_EndOneVParams EQU *

; -----
; Two-bits-per-pixel parameter list
; -----
_TwoBitMode OSLstEntry mVidParams, _TwoVidParams
DatLstEntry mPageCnt, 1
DatLstEntry mDevType, 0 ; 0 is CLUT Type device
DatLstEntry endOfList, 0

_TwoVidParams DC.L _EndTwoVParams-_TwoVidParams ; Physical block size
DC.L kMyBaseOffset ; base offset for start of screen memory
DC.W kMyRowBytes ; rowbytes
DC.W 0,0,480,640 ; bounds (top,left,bottom,right)
DC.W 0 ; version
DC.W 0 ; packType (indicates to use default)
DC.L 0 ; packSize (reserved for future use)
DC.W 72,0 ; horizontal pixels per inch
DC.W 72,0 ; vertical pixels per inch
DC.W 0 ; pixel type is chunky
DC.W 2 ; PixelSize is 2 bits
DC.W 1 ; 1 components
DC.W 2 ; component size is 2 bits
DC.L 0 ; defPlaneBytes
_EndTwoVParams EQU *

; -----
; Four-bits-per-pixel parameter list
; -----
_FourBitMode OSLstEntry mVidParams, _FourVidParams
DatLstEntry mPageCnt, 1
DatLstEntry mDevType, 0 ; 0 is CLUT Type device
DatLstEntry endOfList, 0

_FourVidParams DC.L _EndFourVParams-_FourVidParams ; Physical block size
DC.L kMyBaseOffset ; base offset for start of screen memory
DC.W kMyRowBytes ; rowbytes
DC.W 0,0,480,640 ; bounds (top,left,bottom,right)
DC.W 0 ; version
DC.W 0 ; packType (indicates to use default)
DC.L 0 ; packSize (reserved for future use)
DC.W 72,0 ; horizontal pixels per inch
DC.W 72,0 ; vertical pixels per inch
DC.W 0 ; pixel type is chunky
DC.W 4 ; PixelSize is 4 bits
DC.W 1 ; 1 components in color
DC.W 4 ; component size is 4 bits
DC.L 0 ; defPlaneBytes
_EndFourVParams EQU *

; -----
; Eight-bits-per-pixel parameter list
; -----
_EightBitMode OSLstEntry mVidParams, _EightVidParams
DatLstEntry mPageCnt, 1
DatLstEntry mDevType, 0 ; 0 is CLUT Type device
DatLstEntry endOfList, 0

_EightVidParams DC.L _EndEightVParams-_EightVidParams ; Physical block size
DC.L kMyBaseOffset ; base offset for start of screen memory
DC.W kMyRowBytes ; rowbytes
DC.W 0,0,480,640 ; bounds (top,left,bottom,right)
DC.W 0 ; version
DC.W 0 ; packType (indicates to use default)
DC.L 0 ; packSize (reserved for future use)

```

```

        DC.W      72.0          ; horizontal pixels per inch
        DC.W      72.0          ; vertical pixels per inch
        DC.W      0              ; pixel type is chunky
        DC.W      8              ; PixelSize is 8 bits
        DC.W      1              ; 1 components in color
        DC.W      8              ; component size is 8 bits
        DC.L      0              ; defPlaneBytes
_EndEightVParams EQU      *

; -----
; One-bits-per-pixel parameter list
; -----
_OneVGAMode    OSLstEntry   mVidParams, _OneVGAParams
                DatLstEntry  mPageCnt, 1
                DatLstEntry  mDevType, 0          ; 0 is CLUT Type device
                DatLstEntry  endOfList, 0

_OneVGAParams DC.L       _EndOneVGAParams-_OneVGAParams ; Physical block size
                DC.L       kMyBaseOffset ; base offset for start of screen memory
                DC.W       kMyRowBytes  ; rowbytes
                DC.W       0,0,480,640 ; bounds (top,left,bottom,right)
                DC.W       0              ; version
                DC.W       0              ; packType (indicates to use default)
                DC.L       0              ; packSize (reserved for future use)
                DC.W       72.0          ; horizontal pixels per inch
                DC.W       72.0          ; vertical pixels per inch
                DC.W       0              ; pixel type is chunky
                DC.W       1              ; PixelSize is 1 bits
                DC.W       1              ; 1 component
                DC.W       1              ; component size is 1 bit
                DC.L       0              ; defPlaneBytes
_EndOneVGAParams EQU      *

; -----
; Two-bits-per-pixel parameter list
; -----
_TwoVGAMode    OSLstEntry   mVidParams, _TwoVGAParams
                DatLstEntry  mPageCnt, 1
                DatLstEntry  mDevType, 0          ; 0 is CLUT Type device
                DatLstEntry  endOfList, 0

_TwoVGAParams DC.L       _EndTwoVGAParams-_TwoVGAParams ; Physical block size
                DC.L       kMyBaseOffset ; base offset for start of screen memory
                DC.W       kMyRowBytes  ; rowbytes
                DC.W       0,0,480,640 ; bounds (top,left,bottom,right)
                DC.W       0              ; version
                DC.W       0              ; packType (indicates to use default)
                DC.L       0              ; packSize (reserved for future use)
                DC.W       72.0          ; horizontal pixels per inch
                DC.W       72.0          ; vertical pixels per inch
                DC.W       0              ; pixel type is chunky
                DC.W       2              ; PixelSize is 2 bits
                DC.W       1              ; 1 components
                DC.W       2              ; component size is 2 bits
                DC.L       0              ; defPlaneBytes
_EndTwoVGAParams EQU      *

; -----
; Four-bits-per-pixel parameter list
; -----
_FourVGAMode   OSLstEntry   mVidParams, _FourVGAParams
                DatLstEntry  mPageCnt, 1
                DatLstEntry  mDevType, 0          ; 0 is CLUT Type device
                DatLstEntry  endOfList, 0

_FourVGAParams DC.L       _EndFourVGAParams-_FourVGAParams ; Physical block size
                DC.L       kMyBaseOffset ; base offset for start of screen memory
                DC.W       kMyRowBytes  ; rowbytes
                DC.W       0,0,480,640 ; bounds (top,left,bottom,right)
                DC.W       0              ; version
                DC.W       0              ; packType (indicates to use default)
                DC.L       0              ; packSize (reserved for future use)
                DC.W       72.0          ; horizontal pixels per inch
                DC.W       72.0          ; vertical pixels per inch
                DC.W       0              ; pixel type is chunky
                DC.W       4              ; PixelSize is 4 bits
                DC.W       1              ; 1 components in color
                DC.W       4              ; component size is 4 bits
                DC.L       0              ; defPlaneBytes
_EndFourVGAParams EQU      *

; -----
; Eight-bits-per-pixel parameter list
; -----
_EightVGAMode  OSLstEntry   mVidParams, _EightVGAParams
                DatLstEntry  mPageCnt, 1
                DatLstEntry  mDevType, 0          ; 0 is CLUT Type device
                DatLstEntry  endOfList, 0

```

```

_EightVGAParams DC.L      _EndEightVGAParams-_EightVGAParams; Physical block size
    DC.L          kMyBaseOffset           ; base offset for start of screen memory
    DC.W          kMyRowBytes            ; rowbytes
    DC.W          0,0,480,640           ; bounds (top,left,bottom,right)
    DC.W          0                     ; version
    DC.W          0                     ; packType (indicates to use default)
    DC.L          0                     ; packSize (reserved for future use)
    DC.W          72,0                 ; horizontal pixels per inch
    DC.W          72,0                 ; vertical pixels per inch
    DC.W          0                     ; pixel type is chunky
    DC.W          8                     ; PixelSize is 8 bits
    DC.W          1                     ; 1 components in color
    DC.W          8                     ; component size is 8 bits
    DC.L          0                     ; defPlaneBytes
_EndEightVGAParams EQU      *

; -----
; One-bits-per-pixel parameter list
; -----
_OnePortMode   OSLstEntry     mVidParams, _OnePortParams
    DatLstEntry   mPageCnt, 1
    DatLstEntry   mDevType, 0           ; 0 is CLUT Type device
    DatLstEntry   endOfList, 0

_OnePortParams DC.L      _EndOnePortParams-_OnePortParams           ; Physical block size
    DC.L          kMyBaseOffset           ; base offset for start of screen memory
    DC.W          kMyRowBytes/2          ; rowbytes
    DC.W          0,0,871,640           ; bounds (top,left,bottom,right)
    DC.W          0                     ; version
    DC.L          0                     ; packType (indicates to use default)
    DC.W          72,0                 ; packSize (reserved for future use)
    DC.W          72,0                 ; horizontal pixels per inch
    DC.W          0                     ; vertical pixels per inch
    DC.W          1                     ; pixel type is chunky
    DC.W          1                     ; PixelSize is 1 bits
    DC.W          1                     ; 1 component
    DC.W          1                     ; component size is 1 bit
    DC.L          0                     ; defPlaneBytes
_EndOnePortParams EQU      *

; -----
; Two-bits-per-pixel parameter list
; -----
_TwoPortMode   OSLstEntry     mVidParams, _TwoPortParams
    DatLstEntry   mPageCnt, 1
    DatLstEntry   mDevType, 0           ; 0 is CLUT Type device
    DatLstEntry   endOfList, 0

_TwoPortParams DC.L      _EndTwoPortParams-_TwoPortParams           ; Physical block size
    DC.L          kMyBaseOffset           ; base offset for start of screen memory
    DC.W          kMyRowBytes/2          ; rowbytes
    DC.W          0,0,871,640           ; bounds (top.left,bottom,right)
    DC.W          0                     ; version
    DC.W          0                     ; packType (indicates to use default)
    DC.L          0                     ; packSize (reserved for future use)
    DC.W          72,0                 ; horizontal pixels per inch
    DC.W          72,0                 ; vertical pixels per inch
    DC.W          0                     ; pixel type is chunky
    DC.W          2                     ; PixelSize is 2 bits
    DC.W          1                     ; 1 components
    DC.W          2                     ; component size is 2 bits
    DC.L          0                     ; defPlaneBytes
_EndTwoPortParams EQU      *

; -----
; Four-bits-per-pixel parameter list
; -----
_FourPortMode  OSLstEntry     mVidParams, _FourPortParams
    DatLstEntry   mPageCnt, 1
    DatLstEntry   mDevType, 0           ; 0 is CLUT Type device
    DatLstEntry   endOfList, 0

_FourPortParams DC.L      _EndFourPortParams-_FourPortParams           ; Physical block size
    DC.L          kMyBaseOffset           ; base offset for start of screen memory
    DC.W          kMyRowBytes/2          ; rowbytes
    DC.W          0,0,871,640           ; bounds (top.left,bottom,right)
    DC.W          0                     ; version
    DC.W          0                     ; packType (indicates to use default)
    DC.L          0                     ; packSize (reserved for future use)
    DC.W          72,0                 ; horizontal pixels per inch
    DC.W          72,0                 ; vertical pixels per inch
    DC.W          0                     ; pixel type is chunky
    DC.W          4                     ; PixelSize is 4 bits
    DC.W          1                     ; 1 components in color
    DC.W          4                     ; component size is 4 bits
    DC.L          0                     ; defPlaneBytes
_EndFourPortParams EQU      *

```

```

; One-bits-per-pixel parameter list
; -----
_OneBartMode OSListEntry mVidParams, _OneBartParams
DatListEntry mPageCnt, 1
DatListEntry mDevType, 0 ; 0 is CLUT Type device
DatListEntry endOfList, 0

_OneBartParams DC.L _EndOneBartParams-_OneBartParams ; Physical block size
DC.L kMyBaseOffset ; base offset for start of screen memory
DC.W kMyRowBytes ; rowbytes
DC.W 0,0,342,512 ; bounds (top,left,bottom,right)
DC.W 0 ; version
DC.W 0 ; packType (indicates to use default)
DC.L 0 ; packSize (reserved for future use)
DC.W 72,0 ; horizontal pixels per inch
DC.W 72,0 ; vertical pixels per inch
DC.W 0 ; pixel type is chunky
DC.W 1 ; PixelSize is 1 bits
DC.W 1 ; 1 component
DC.W 1 ; component size is 1 bit
DC.L 0 ; defPlaneBytes
_EndOneBartParams EQU *

; Two-bits-per-pixel parameter list
; -----
_TwoBartMode OSListEntry mVidParams, _TwoBartParams
DatListEntry mPageCnt, 1
DatListEntry mDevType, 0 ; 0 is CLUT Type device
DatListEntry endOfList, 0

_TwoBartParams DC.L _EndTwoBartParams-_TwoBartParams ; Physical block size
DC.L kMyBaseOffset ; base offset for start of screen memory
DC.W kMyRowBytes ; rowbytes
DC.W 0,0,342,512 ; bounds (top,left,bottom,right)
DC.W 0 ; version
DC.W 0 ; packType (indicates to use default)
DC.L 0 ; packSize (reserved for future use)
DC.W 72,0 ; horizontal pixels per inch
DC.W 72,0 ; vertical pixels per inch
DC.W 0 ; pixel type is chunky
DC.W 2 ; PixelSize is 2 bits
DC.W 1 ; 1 components
DC.W 2 ; component size is 2 bits
DC.L 0 ; defPlaneBytes
_EndTwoBartParams EQU *

; Four-bits-per-pixel parameter list
; -----
_FourBartMode OSListEntry mVidParams, _FourBartParams
DatListEntry mPageCnt, 1
DatListEntry mDevType, 0 ; 0 is CLUT Type device
DatListEntry endOfList, 0

_FourBartParams DC.L _EndFourBartParams-_FourBartParams ; Physical block size
DC.L kMyBaseOffset ; base offset for start of screen memory
DC.W kMyRowBytes ; rowbytes
DC.W 0,0,342,512 ; bounds (top,left,bottom,right)
DC.W 0 ; version
DC.W 0 ; packType (indicates to use default)
DC.L 0 ; packSize (reserved for future use)
DC.W 72,0 ; horizontal pixels per inch
DC.W 72,0 ; vertical pixels per inch
DC.W 0 ; pixel type is chunky
DC.W 4 ; PixelSize is 4 bits
DC.W 1 ; 1 components in color
DC.W 4 ; component size is 4 bits
DC.L 0 ; defPlaneBytes
_EndFourBartParams EQU *

; Eight-bits-per-pixel parameter list
; -----
_EightBartMode OSListEntry mVidParams, _EightBartParams
DatListEntry mPageCnt, 1
DatListEntry mDevType, 0 ; 0 is CLUT Type device
DatListEntry endOfList, 0

_EightBartParams DC.L _EndEightBartParams-_EightBartParams ; Physical block size
DC.L kMyBaseOffset ; base offset for start of screen memory
DC.W kMyRowBytes ; rowbytes
DC.W 0,0,342,512 ; bounds (top,left,bottom,right)
DC.W 0 ; version
DC.W 0 ; packType (indicates to use default)
DC.L 0 ; packSize (reserved for future use)

```

```

        DC.W      72,0          ; horizontal pixels per inch
        DC.W      72,0          ; vertical pixels per inch
        DC.W      0             ; pixel type is chunky
        DC.W      8             ; PixelSize is 8 bits
        DC.W      1             ; 1 components in color
        DC.W      8             ; component size is 8 bits
        DC.L      0             ; defPlaneBytes
_EndEightBartParams EQU    *

; -----
; One-bits-per-pixel parameter list
;
_OneV1kx1kMode OSLstEntry   mVidParams, _OneV1kx1kParams
        DatLstEntry   mPageCnt, 1
        DatLstEntry   mDevType, 0           ; 0 is CLUT Type device
        DatLstEntry   endOfList, 0

_OneV1kx1kParams DC.L      _EndOneV1kx1kVParams-_OneV1kx1kParams ; Physical block size
        DC.L      kMyBaseOffset          ; base offset for start of screen memory
        DC.W      kMyRowBytes/2          ; rowbytes
        DC.W      0,0,1023,1024         ; bounds (top,left,bottom,right)
        DC.W      0             ; version
        DC.W      0             ; packType (indicates to use default)
        DC.L      0             ; packSize (reserved for future use)
        DC.W      72,0          ; horizontal pixels per inch
        DC.W      72,0          ; vertical pixels per inch
        DC.W      0             ; pixel type is chunky
        DC.W      1             ; PixelSize is 1 bits
        DC.W      1             ; 1 component
        DC.W      1             ; component size is 1 bit
        DC.L      0             ; defPlaneBytes
_EndOneV1kx1kVParams EQU    *

; -----
; Two-bits-per-pixel parameter list
;
_TwoV1kx1kMode OSLstEntry   mVidParams, _TwoV1kx1kParams
        DatLstEntry   mPageCnt, 1
        DatLstEntry   mDevType, 0           ; 0 is CLUT Type device
        DatLstEntry   endOfList, 0

_TwoV1kx1kParams DC.L      _EndTwoV1kx1kVParams-_TwoV1kx1kParams ; Physical block size
        DC.L      kMyBaseOffset          ; base offset for start of screen memory
        DC.W      kMyRowBytes/2          ; rowbytes
        DC.W      0,0,1023,1024         ; bounds (top,left,bottom,right)
        DC.W      0             ; version
        DC.W      0             ; packType (indicates to use default)
        DC.L      0             ; packSize (reserved for future use)
        DC.W      72,0          ; horizontal pixels per inch
        DC.W      72,0          ; vertical pixels per inch
        DC.W      0             ; pixel type is chunky
        DC.W      2             ; PixelSize is 2 bits
        DC.W      1             ; 1 components
        DC.W      2             ; component size is 2 bits
        DC.L      0             ; defPlaneBytes
_EndTwoV1kx1kVParams EQU    *

; -----
; Four-bits-per-pixel parameter list
;
_FourV1kx1kMode OSLstEntry  mVidParams, _FourV1kx1kParams
        DatLstEntry   mPageCnt, 1
        DatLstEntry   mDevType, 0           ; 0 is CLUT Type device
        DatLstEntry   endOfList, 0

_FourV1kx1kParams DC.L      _EndFourV1kx1kVParams-_FourV1kx1kParams ; Physical block size
        DC.L      kMyBaseOffset          ; base offset for start of screen memory
        DC.W      kMyRowBytes/2          ; rowbytes
        DC.W      0,0,1023,1024         ; bounds (top,left,bottom,right)
        DC.W      0             ; version
        DC.W      0             ; packType (indicates to use default)
        DC.L      0             ; packSize (reserved for future use)
        DC.W      72,0          ; horizontal pixels per inch
        DC.W      72,0          ; vertical pixels per inch
        DC.W      0             ; pixel type is chunky
        DC.W      4             ; PixelSize is 4 bits
        DC.W      1             ; 1 components in color
        DC.W      4             ; component size is 4 bits
        DC.L      0             ; defPlaneBytes
_EndFourV1kx1kVParams EQU    *

; -----
; One-bits-per-pixel parameter list
;
_OneV1kx512Mode OSLstEntry  mVidParams, _OneV1kx512Params
        DatLstEntry   mPageCnt, 1
        DatLstEntry   mDevType, 0           ; 0 is CLUT Type device
        DatLstEntry   endOfList, 0

```

```

_OneV1kx512Params DC.L      _EndOneV1kx512VParams-_OneV1kx512Params ; Physical block size
.                 DC.L      kMyBaseOffset          ; base offset for start of screen memory
DC.W              kMyRowBytes           ; rowbytes
DC.W              0,0,511,1024        ; bounds (top,left,bottom,right)
DC.W              0                   ; version
DC.W              0                   ; packType (indicates to use default)
DC.L              0                   ; packSize (reserved for future use)
DC.W              72,0                ; horizontal pixels per inch
DC.W              72,0                ; vertical pixels per inch
DC.W              0                   ; pixel type is chunky
DC.W              1                   ; PixelSize is 1 bits
DC.W              1                   ; 1 component
DC.W              1                   ; component size is 1 bit
DC.L              0                   ; defPlaneBytes
_EndOneV1kx512VParams EQU     *

; -----
; Two-bits-per-pixel parameter list
; -----
_TwoV1kx512Mode OSLstEntry    mVidParams, _TwoV1kx512Params
DatLstEntry       mPageCnt, 1
DatLstEntry       mDevType, 0           ; 0 is CLUT Type device
DatLstEntry       endOfList, 0

_TwoV1kx512Params DC.L      _EndTwoV1kx512VParams-_TwoV1kx512Params ; Physical block size
.                 DC.L      kMyBaseOffset          ; base offset for start of screen memory
DC.W              kMyRowBytes           ; rowbytes
DC.W              0,0,511,1024        ; bounds (top,left,bottom,right)
DC.W              0                   ; version
DC.L              0                   ; packType (indicates to use default)
DC.W              72,0                ; packSize (reserved for future use)
DC.W              72,0                ; horizontal pixels per inch
DC.W              72,0                ; vertical pixels per inch
DC.W              0                   ; pixel type is chunky
DC.W              2                   ; PixelSize is 2 bits
DC.W              1                   ; 1 components
DC.W              2                   ; component size is 2 bits
DC.L              0                   ; defPlaneBytes
_EndTwoV1kx512VParams EQU     *

; -----
; Four-bits-per-pixel parameter list
; -----
_FourV1kx512Mode OSLstEntry   mVidParams, _FourV1kx512Params
DatLstEntry       mPageCnt, 1
DatLstEntry       mDevType, 0           ; 0 is CLUT Type device
DatLstEntry       endOfList, 0

_FourV1kx512Params DC.L      _EndFourV1kx512VParams-_FourV1kx512Params ; Physical block size
.                 DC.L      kMyBaseOffset          ; base offset for start of screen memory
DC.W              kMyRowBytes           ; rowbytes
DC.W              0,0,511,1024        ; bounds (top,left,bottom,right)
DC.W              0                   ; version
DC.W              0                   ; packType (indicates to use default)
DC.L              0                   ; packSize (reserved for future use)
DC.W              72,0                ; horizontal pixels per inch
DC.W              72,0                ; vertical pixels per inch
DC.W              0                   ; pixel type is chunky
DC.W              4                   ; PixelSize is 4 bits
DC.W              1                   ; 1 components in color
DC.W              4                   ; component size is 4 bits
DC.L              0                   ; defPlaneBytes
_EndFourV1kx512VParams EQU     *

; -----
; Eight-bits-per-pixel parameter list
; -----
_EightV1kx512Mode OSLstEntry  mVidParams, _EightV1kx512Params
DatLstEntry       mPageCnt, 1
DatLstEntry       mDevType, 0           ; 0 is CLUT Type device
DatLstEntry       endOfList, 0

_EightV1kx512Params DC.L      _EndEightV1kx512VParams-_EightV1kx512Params ; Physical block size
.                DC.L      kMyBaseOffset          ; base offset for start of screen memory
DC.W             kMyRowBytes           ; rowbytes
DC.W             0,0,511,1024        ; bounds (top,left,bottom,right)
DC.W             0                   ; version
DC.W             0                   ; packType (indicates to use default)
DC.L             0                   ; packSize (reserved for future use)
DC.W             72,0                ; horizontal pixels per inch
DC.W             72,0                ; vertical pixels per inch
DC.W             0                   ; pixel type is chunky
DC.W             8                   ; PixelSize is 4 bits
DC.W             1                   ; 1 components in color
DC.W             8                   ; component size is 4 bits
DC.L             0                   ; defPlaneBytes
_EndEightV1kx512VParams EQU     *

```

```

; -----
;rb One-bits-per-pixel parameter list  new rez
; -----
_OneTweMode    OSLstEntry      mVidParams, _OneTweParams
                DatLstEntry     mPageCnt, 1
                DatLstEntry     mDevType, 0          ; 0 is CLUT Type device
                DatLstEntry     endOfList, 0

_OneTweParams  DC.L           _EndOneTweParams-_OneTweParams ; Physical block size
                DC.L           kMyBaseOffset   ; base offset for start of screen memory
                DC.W           kMyRowBytes    ; rowbytes
                DC.W           0,0,384,512   ; bounds (top,left,bottom,right)
                DC.W           0             ; version
                DC.W           0             ; packType (indicates to use default)
                DC.L           0             ; packSize (reserved for future use)
                DC.W           72,0          ; horizontal pixels per inch
                DC.W           72,0          ; vertical pixels per inch
                DC.W           0             ; pixel type is chunky
                DC.W           1             ; PixelSize is 1 bits
                DC.W           1             ; 1 component
                DC.W           1             ; component size is 1 bit
                DC.L           0             ; defPlaneBytes

_EndOneTweParams EQU        *

; -----
;rb Two-bits-per-pixel parameter list  new rez
; -----
_TwoTweMode    OSLstEntry      mVidParams, _TwoTweParams
                DatLstEntry     mPageCnt, 1
                DatLstEntry     mDevType, 0          ; 0 is CLUT Type device
                DatLstEntry     endOfList, 0

_TwoTweParams  DC.L           _EndTwoTweParams-_TwoTweParams ; Physical block size
                DC.L           kMyBaseOffset   ; base offset for start of screen memory
                DC.W           kMyRowBytes    ; rowbytes
                DC.W           0,0,384,512   ; bounds (top,left,bottom,right)
                DC.W           0             ; version
                DC.W           0             ; packType (indicates to use default)
                DC.L           0             ; packSize (reserved for future use)
                DC.W           72,0          ; horizontal pixels per inch
                DC.W           72,0          ; vertical pixels per inch
                DC.W           0             ; pixel type is chunky
                DC.W           2             ; PixelSize is 2 bits
                DC.W           1             ; 1 components
                DC.W           2             ; component size is 2 bits
                DC.L           0             ; defPlaneBytes

_EndTwoTweParams EQU        *

; -----
;rb Four-bits-per-pixel parameter list  for the new rez
; -----
_FourTweMode   OSLstEntry     mVidParams, _FourTweParams
                DatLstEntry     mPageCnt, 1
                DatLstEntry     mDevType, 0          ; 0 is CLUT Type device
                DatLstEntry     endOfList, 0

_FourTweParams DC.L          _EndFourTweParams-_FourTweParams ; Physical block size
                DC.L           kMyBaseOffset   ; base offset for start of screen memory
                DC.W           kMyRowBytes    ; rowbytes
                DC.W           0,0,384,512   ; bounds (top,left,bottom,right)
                DC.W           0             ; version
                DC.W           0             ; packType (indicates to use default)
                DC.L           0             ; packSize (reserved for future use)
                DC.W           72,0          ; horizontal pixels per inch
                DC.W           72,0          ; vertical pixels per inch
                DC.W           0             ; pixel type is chunky
                DC.W           4             ; PixelSize is 4 bits
                DC.W           1             ; 1 components in color
                DC.W           4             ; component size is 4 bits
                DC.L           0             ; defPlaneBytes

_EndFourTweParams EQU        *

; -----
; Eight-bits-per-pixel parameter list
; -----
_EightTweMode  OSLstEntry    mVidParams, _EightTweParams
                DatLstEntry    mPageCnt, 1
                DatLstEntry    mDevType, 0          ; 0 is CLUT Type device
                DatLstEntry    endOfList, 0

_EightTweParams DC.L         _EndEightTweParams-_EightTweParams ; Physical block size
                DC.L          kMyBaseOffset   ; base offset for start of screen memory
                DC.W          kMyRowBytes    ; rowbytes
                DC.W          0,0,384,512   ; bounds (top,left,bottom,right)
                DC.W          0             ; version
                DC.W          0             ; packType (indicates to use default)
                DC.L          0             ; packSize (reserved for future use)

```

```

        DC.W      72,0          ; horizontal pixels per inch
        DC.W      72,0          ; vertical pixels per inch
        DC.W      0             ; pixel type is chunky
        DC.W      8             ; PixelSize is 4 bits
        DC.W      1             ; 1 components in color
        DC.W      8             ; component size is 4 bits
        DC.L      0             ; defPlaneBytes
_EndEightTweParams EQU      *

; -----
;rb One-bits-per-pixel parameter list  new rez
;
_One8x6Mode   OSLstEntry    mVidParams, _One8x6Params
               DatLstEntry   mPageCnt, 1
               DatLstEntry   mDevType, 0           ; 0 is CLUT Type device
               DatLstEntry   endOfList, 0

_One8x6Params DC.L      _EndOne8x6Params-_One8x6Params ; Physical block size
               DC.L      kMyBaseOffset          ; base offset for start of screen memory
               DC.W      kMyRowBytes/2         ; rowbytes
               DC.W      0,0,600,800          ; bounds (top,left,bottom,right)
               DC.W      0             ; version
               DC.W      0             ; packType (indicates to use default)
               DC.L      0             ; packSize (reserved for future use)
               DC.W      72,0          ; horizontal pixels per inch
               DC.W      72,0          ; vertical pixels per inch
               DC.W      0             ; pixel type is chunky
               DC.W      1             ; PixelSize is 1 bits
               DC.W      1             ; 1 component
               DC.W      1             ; component size is 1 bit
               DC.L      0             ; defPlaneBytes
_EndOne8x6Params EQU      *

; -----
;rb Two-bits-per-pixel parameter list  new rez
;
_Two8x6Mode   OSLstEntry    mVidParams, _Two8x6Params
               DatLstEntry   mPageCnt, 1
               DatLstEntry   mDevType, 0           ; 0 is CLUT Type device
               DatLstEntry   endOfList, 0

_Two8x6Params DC.L      _EndTwo8x6Params-_Two8x6Params ; Physical block size
               DC.L      kMyBaseOffset          ; base offset for start of screen memory
               DC.W      kMyRowBytes/2         ; rowbytes
               DC.W      0,0,600,800          ; bounds (top,left,bottom,right)
               DC.W      0             ; version
               DC.W      0             ; packType (indicates to use default)
               DC.L      0             ; packSize (reserved for future use)
               DC.W      72,0          ; horizontal pixels per inch
               DC.W      72,0          ; vertical pixels per inch
               DC.W      0             ; pixel type is chunky
               DC.W      2             ; PixelSize is 2 bits
               DC.W      1             ; 1 components
               DC.W      2             ; component size is 2 bits
               DC.L      0             ; defPlaneBytes
_EndTwo8x6Params EQU      *

; -----
;rb Four-bits-per-pixel parameter list  for the new rez
;
_Four8x6Mode  OSLstEntry   mVidParams, _Four8x6Params
               DatLstEntry  mPageCnt, 1
               DatLstEntry  mDevType, 0           ; 0 is CLUT Type device
               DatLstEntry  endOfList, 0

_Four8x6Params DC.L      _EndFour8x6Params-_Four8x6Params ; Physical block size
               DC.L      kMyBaseOffset          ; base offset for start of screen memory
               DC.W      kMyRowBytes/2         ; rowbytes
               DC.W      0,0,600,800          ; bounds (top,left,bottom,right)
               DC.W      0             ; version
               DC.W      0             ; packType (indicates to use default)
               DC.L      0             ; packSize (reserved for future use)
               DC.W      72,0          ; horizontal pixels per inch
               DC.W      72,0          ; vertical pixels per inch
               DC.W      0             ; pixel type is chunky
               DC.W      4             ; PixelSize is 4 bits
               DC.W      1             ; 1 components in color
               DC.W      4             ; component size is 4 bits
               DC.L      0             ; defPlaneBytes
_EndFour8x6Params EQU      *

; -----
;rb One-bits-per-pixel parameter list  new rez
;
_One832x624Mode OSLstEntry  mVidParams, _One832x624Params
                  DatLstEntry  mPageCnt, 1
                  DatLstEntry  mDevType, 0           ; 0 is CLUT Type device
                  DatLstEntry  endOfList, 0

```

```

_One832x624Params DC.L _EndOne832x624Params-_One832x624Params ; Physical block size
    . DC.L kMyBaseOffset ; base offset for start of screen memory
    DC.W kMyRowBytes/2 ; rowbytes
    DC.W 0,0,624,832 ; bounds (top,left,bottom,right)
    DC.W 0 ; version
    DC.W 0 ; packType (indicates to use default)
    DC.L 0 ; packSize (reserved for future use)
    DC.W 72,0 ; horizontal pixels per inch
    DC.W 72,0 ; vertical pixels per inch
    DC.W 0 ; pixel type is chunky
    DC.W 1 ; PixelSize is 1 bits
    DC.W 1 ; 1 component
    DC.W 1 ; component size is 1 bit
    DC.L 0 ; defPlaneBytes
_EndOne832x624Params EQU *

; -----
;rb Two-bits-per-pixel parameter list new rez
; -----
_Two832x624Mode OSLstEntry mVidParams, _Two832x624Params
    DatLstEntry mPageCnt, 1
    DatLstEntry mDevType, 0 ; 0 is CLUT Type device
    DatLstEntry endOfList, 0

_Two832x624Params DC.L _EndTwo832x624Params-_Two832x624Params ; Physical block size
    DC.L kMyBaseOffset ; base offset for start of screen memory
    DC.W kMyRowBytes/2 ; rowbytes
    DC.W 0,0,624,832 ; bounds (top,left,bottom,right)
    DC.W 0 ; version
    DC.L 0 ; packType (indicates to use default)
    DC.W 72,0 ; packSize (reserved for future use)
    DC.W 72,0 ; horizontal pixels per inch
    DC.W 72,0 ; vertical pixels per inch
    DC.W 0 ; pixel type is chunky
    DC.W 2 ; PixelSize is 2 bits
    DC.W 1 ; 1 components
    DC.W 2 ; component size is 2 bits
    DC.L 0 ; defPlaneBytes
_EndTwo832x624Params EQU *

; -----
;rb Four-bits-per-pixel parameter list for the new rez
; -----
_Four832x624Mode OSLstEntry mVidParams, _Four832x624Params
    DatLstEntry mPageCnt, 1
    DatLstEntry mDevType, 0 ; 0 is CLUT Type device
    DatLstEntry endOfList, 0

_Four832x624Params DC.L _EndFour832x624Params-_Four832x624Params ; Physical block size
    DC.L kMyBaseOffset ; base offset for start of screen memory
    DC.W kMyRowBytes/2 ; rowbytes
    DC.W 0,0,624,832 ; bounds (top,left,bottom,right)
    DC.W 0 ; version
    DC.L 0 ; packType (indicates to use default)
    DC.W 72,0 ; packSize (reserved for future use)
    DC.W 72,0 ; horizontal pixels per inch
    DC.W 72,0 ; vertical pixels per inch
    DC.W 0 ; pixel type is chunky
    DC.W 4 ; PixelSize is 4 bits
    DC.W 1 ; 1 components in color
    DC.W 4 ; component size is 4 bits
    DC.L 0 ; defPlaneBytes
_EndFour832x624Params EQU *

; -----
;rb NOTE 32 One-bits-per-pixel parameter list new rez
; -----
_One1x7Mode OSLstEntry mVidParams, _One1x7Params
    DatLstEntry mPageCnt, 1
    DatLstEntry mDevType, 0 ; 0 is CLUT Type device
    DatLstEntry endOfList, 0

_One1x7Params DC.L _EndOne1x7Params-_One1x7Params ; Physical block size
    DC.L kMyBaseOffset ; base offset for start of screen memory
    DC.W kMyRowBytes/2 ; rowbytes
    DC.W 0,0,768,1024 ; bounds (top,left,bottom,right)
    DC.W 0 ; version
    DC.W 0 ; packType (indicates to use default)
    DC.L 0 ; packSize (reserved for future use)
    DC.W 72,0 ; horizontal pixels per inch
    DC.W 72,0 ; vertical pixels per inch
    DC.W 0 ; pixel type is chunky
    DC.W 1 ; PixelSize is 1 bits
    DC.W 1 ; 1 component
    DC.W 1 ; component size is 1 bit
    DC.L 0 ; defPlaneBytes
_EndOne1x7Params EQU *

```

```

;rb Two-bits-per-pixel parameter list  new rez
;
_Twolx7Mode    OSLstEntry      mVidParams, _Twolx7Params
                DatLstEntry     mPageCnt, 1
                DatLstEntry     mDevType, 0
                DatLstEntry     endOfList, 0 ; 0 is CLUT Type device

_Twolx7Params  DC.L           _EndTwolx7Params-_Twolx7Params ; Physical block size
                DC.L           kMyBaseOffset ; base offset for start of screen memory
                DC.W           kMyRowBytes/2 ; rowbytes
                DC.W           0,0,768,1024 ; bounds (top,left,bottom,right)
                DC.W           0 ; version
                DC.W           0 ; packType (indicates to use default)
                DC.L           0 ; packSize (reserved for future use)
                DC.W           72,0 ; horizontal pixels per inch
                DC.W           72,0 ; vertical pixels per inch
                DC.W           0 ; pixel type is chunky
                DC.W           2 ; PixelSize is 2 bits
                DC.W           1 ; 1 components
                DC.W           2 ; component size is 2 bits
                DC.L           0 ; defPlaneBytes

_EndTwolx7Params EQU          *
;

;rb Four-bits-per-pixel parameter list  for the new rez
;
_Fourlx7Mode   OSLstEntry      mVidParams, _Fourlx7Params
                DatLstEntry     mPageCnt, 1
                DatLstEntry     mDevType, 0 ; 0 is CLUT Type device
                DatLstEntry     endOfList, 0

_Fourlx7Params DC.L           _EndFourlx7Params-_Fourlx7Params ; Physical block size
                DC.L           kMyBaseOffset ; base offset for start of screen memory
                DC.W           kMyRowBytes/2 ; rowbytes
                DC.W           0,0,768,1024 ; bounds (top,left,bottom,right)
                DC.W           0 ; version
                DC.W           0 ; packType (indicates to use default)
                DC.L           0 ; packSize (reserved for future use)
                DC.W           72,0 ; horizontal pixels per inch
                DC.W           72,0 ; vertical pixels per inch
                DC.W           0 ; pixel type is chunky
                DC.W           4 ; PixelSize is 4 bits
                DC.W           1 ; 1 components in color
                DC.W           4 ; component size is 4 bits
                DC.L           0 ; defPlaneBytes

_EndFourlx7Params EQU          *
;

; Fill memory from here to Header Block at end of ROM with $FF
;
DCB.B          (kROMSize-191)-(*-VideoDeclROM), $FF

;
; FORMAT/HEADER BLOCK
;
ORG            kROMSize - 208; fhBlock(fhBlockSize)

;
;rb NOTE 26 Resolution Table
;

STRING PASCAL
DC.B           'Apple 12inch ',0 ;rb NOTE 36
DC.B           134
DC.B           'Apple 16inch ',0 ;rb each string must be 14 char
DC.B           138
DC.B           'Apple Portrait',0
DC.B           130
DC.B           '640x480      ',0
DC.B           128
DC.B           '640x480 (VGA)',0
DC.B           129
DC.B           '800x600 (56Hz)',0
DC.B           135
DC.B           '800x600 (60Hz)',0
DC.B           136
DC.B           '800x600 (72Hz)',0
DC.B           137
DC.B           '832x624      ',0
DC.B           138
DC.B           '1024x768     ',0
DC.B           139
DC.B           'Internal     ',0
DC.B           131
DC.B           11 ;rb this is how many there is

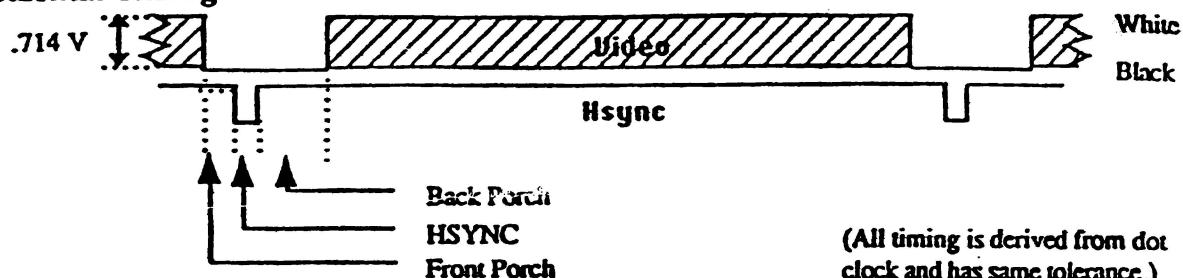
```

# PRELIMINARY SPEC

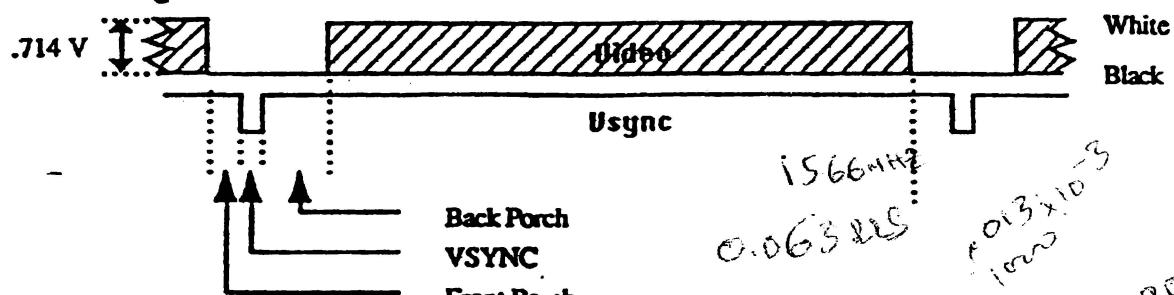
16" MONITOR

## 5.1 Input signal timing

### Horizontal Timing



### Vertical Timing



active pixels per line	832	active # of lines	624
interlace ratio	1.00	vertical refresh (Hz)	75.00
vertical blank lines	30.00	Total lines	654.00
		line time (us)	: 20.39 → 323.65
vertical front porch (lines)	3	vertical blanking time (us)	611.62 → 9709.79
vertical sync (lines)	3	time (us)	61.16 → 970.79
vertical back porch (lines)	24	time (us)	61.16 → 970.79
Horiz blank words	18	time (us)	489.30 → 7766.66
H front porch words	2	Total V retrace	550.46 → 20
H sync words	4	Horiz blank pixels	288
H back porch words	12	Horiz blank words	18.00
		Horiz blanking time (us)	5.24 → 83.17
pixel frequency (Hz)	54,936,000	Horiz frequency (Hz)	49,050.00 → 319.26
		H front porch pixels	32.00
		H sync pixels	64.00
		H back porch pixels	192.00
		H front porch time (ns)	582.50 → 9.2 → 1.164.99 → 18.49
		H sync time (ns)	1.164.99 → 18.49 → 3.494.98 → 55.47
		H back porch time (ns)	3.494.98 → 55.47 → 4,659.97 → 73.96
		Total H retrace	4,659.97 → 73.96
		words / line	70.00
		time / word (ns)	291.25 → 4.62
		pixel time (ns)	18.20 → 17.485 → 0.25

66  
APPLE CONFIDENTIAL

20.39 μs

582.50

$\frac{19.2}{7} = 17.485 \mu s$   
 $\frac{19.2}{7} = 17.485 \mu s$

31 30 29 28 27 26 25 24 23 22 21 20  
31 - 28 27 - 24 23 - 20 19 - 16 15 - 12 11 - 8 7 - 4 3 - 0  
215 666 1972  
TAXIING O COMPANY O 22

IR-25-21 THU

24

A large rectangle is divided into four quadrants by a horizontal line near the top and a vertical line near the left side. The bottom-left quadrant contains the handwritten text "1971 NOV 21".

28 dot pitch  
flat screen 17 "

Company  
Confidential

A small, roughly oval-shaped sketch located in the bottom right corner of the page. It contains some internal, illegible markings.

<b>ICS PART NUMBER (Status)</b>	<b>ICS1494-536 (In Prod)</b>
<b>1394 Equiv</b>	
<b>CUSTOMER</b>	<b>MICRON TECH 20-0319</b>
<b>COMPATIBLE VGA CHIPSET</b>	
<b>ADDRESS LOCATION (Hex)</b>	<b>OUTPUT FREQUENCY (MHz)</b>
00	30.240
01	15.667
02	17.234
03	12.273
04	57.283
05	80.000
06	100.000
07	EXTFREQ
08	20.000
09	25.175
0A	28.322
0B	31.320
0C	34.000
0D	36.000
0E	38.000
0F	40.000
10	44.900
11	48.000
12	50.350
13	54.000
14	60.000
15	65.000
16	70.000
17	75.000
18	85.000
19	90.000
1A	95.000
1B	105.000
1C	110.000
1D	115.000
1E	120.000
1F	125.000

**NOTES:** All patterns tagged "STD" are to be considered as custom patterns, not

## MAVERICK MEMORY MAP AND MODE REGISTER DESCRIPTIONS

RAM Fs00 0000 - Fs7F FFFF	Mode A Fs80 0000 - Fs9F 0000	Mode B/External Modes FaA0 0000 - FaBF FFFF	DAC FaC0 0000 - FaDF FFFF	ROM FaE0 0000 - FaFF FFFF
------------------------------	---------------------------------	--	------------------------------	------------------------------

Register Address	Contents	Bit #	Function
Fs800000	Horiz Front Porch	0-6	Horizontal Front Porch Interval
Fs800004	HSync/2	0-5	Half of the Horizontal Sync pulse width
Fs800008	Horiz Back Porch	0-10	Horizontal back porch interval - NS
Fs80000C	Line2 (HLINE)	0-10	[# of visible pixels - (FS1 + FS2)] / 2
Fs800010	Vert Front Porch	0-4	Vertical front porch interval
Fs800014	Vert Sync	0-6	Vertical sync interval
Fs800018	Vert Back Porch	0-10	Vertical back porch interval
Fs80001C	Lines/Frame	0-10	Number of visible lines per frame or field
Fs800020	HSync22	0-10	FS1 + FS2 + NS + Horiz Front Porch + Horiz back porch + Line/2
Fs800024	Row MSBs	0-10	Row transfer address MSBs
Fs800028	Row LSB	9	Row transfer address LSB
Fs800028	Col / Row LSB	0-8	Column transfer address
Fs80002C	Mode Reg A	0	Equalization pulse Disable (1 = no equalization pulses)
Fs80002C	Mode Reg A	1	Interface mode enable
Fs80002C	Mode Reg A	2	External Sync Mode Disable (0 = external sync mode)
Fs80002C	Mode Reg A	3,4,5	ROW/COL transfer address format
Fs80002C	Mode Reg A	6	spare
Fs80002C	Mode Reg A	7	G300 Disable (0 = G300 ENABLED)
Fs80002C	Mode Reg A	8,9	Number of Column banks (0 = 1 bank, ..., 3 = 4 banks)
Fs80002C	Mode Reg A	10	Vertical count by whollines
Fs800030	FS1	0-5	Format run pipe fill
Fs800034	FS2	0-5	Format run off
Fs800038	Req to Trans Dly	0-10	Seamless transfer request to transfer delay count
Fs80003C	Trans to Req Dly	0-10	Seamless transfer to next request count
FaA00000	External Mode	0-3	External mode bits that go off chip
FaA00004	Refresh Count	0-4	Transfer cycle start to transfer cycle delay for refresh cycles
FaA00008	Mode Reg B	0-8	video on   Interrupts on   NuBus block mode transfer enable   Debug D3   2 or 4 meg Row banks
FaA00008	Mode Reg B	0	NoSnow Enable   Seamless   VRAM Split transfers   FIDsel   DebugSel (2)
FaA00008	Mode Reg B	1	Video timing enable
FaA00008	Mode Reg B	2	Interrupt enable
FaA00008	Mode Reg B	3	NuBus Block Transfer enable
FaA00008	Mode Reg B	4	Debug data
FaA00008	Mode Reg B	5	Row bank size (0 = 2meg, 1 = 4meg)
FaA00008	Mode Reg B	6	No Snow enable
FaA00008	Mode Reg B	7	Seamless
FaA00008	Mode Reg B	8	Split
FaA00008	Mode Reg B	9,10	FIDsel
FaA00008	Mode Reg B	9,10	DebugSel (0 = TRANSWAITB, 1 = DRAM, 2 = CYCLE1, 3 = Debug data)
FaA0000C	Zoom	0-4	Zoom factor
FaA00010	NS	0-7	Period in Horizontal back porch interval to begin holding off DAC accesses to prevent screen 'snow'
FaA00014	Interrupt Clear	X	This address is written to for clearing the interrupts
FaA00018	External		
FaA0001C	External		
FaA00020	External		
FaA00024	External		
FaA00028	External		
FaA0002C	External		
FaA00030	External		
FaA00034	External		
FaA00038	External		
FaA0003C	External		

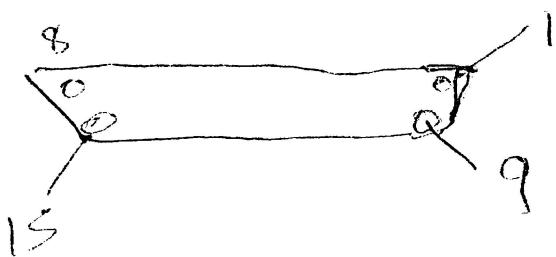
31.625 MHz

20.54

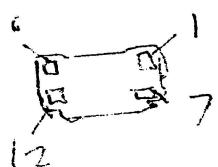
12.03

(0.031us)

# SE/30 Video Cable



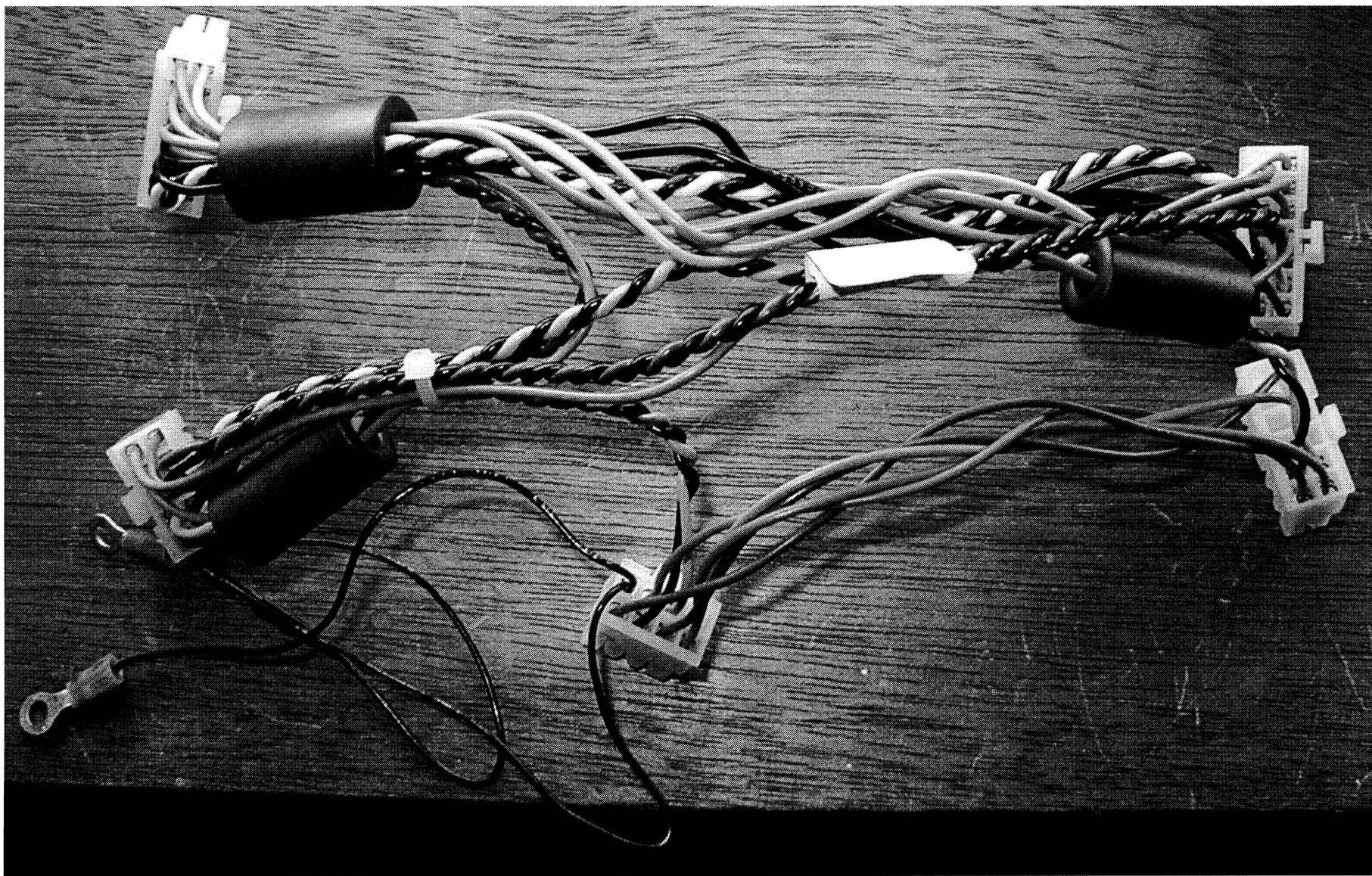
1 Black	11 Black
2 Red1	12 Orange
3 Brown	13 Black
4 N/C	14 Black
5 Green	15 White + Red
6 Black	
7 N/C	
8 N/C	
9 Yellow	
10 Red2	



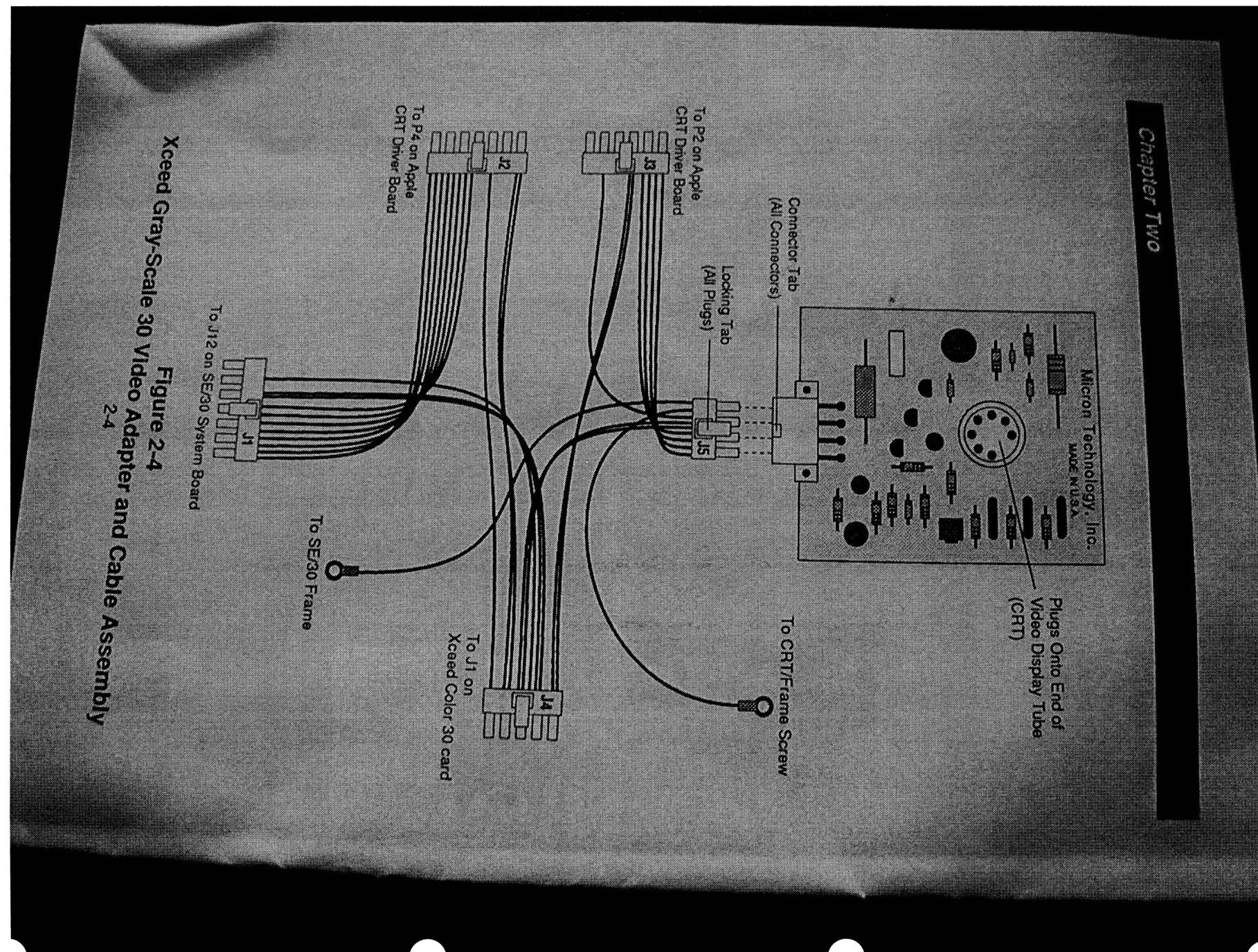
1 Red2	7 N/C
2 Yellow	8 Black
3 Green	9 Black
4 Red1	10 Black
5 Brown	11 Black
6 Orange	12 white + red

harness.JPG (JPEG Image, 1435x903 pixels)

<http://www.eskimo.com/~dslarsen/Micron/harness.JPG>



## Chapter Two



Xceed Gray-Scale 30 Video Adapter and Cable Assembly  
Figure 2-4  
2-4

**Micron SE/30 GrayScale Adaptor**Sponsored by [JUNKYARD JEFFREY](#)Credit to D.S. Larsen's [images](#).Credit to [Mark Janello](#)**THIS GRayscale ADAPTOR WORKS ONLY WITH Xceed Color 30™, Color 30HR™, or MacroColor 30™ MICRON VIDEO CARDS.**

This web page attempts to provide enough information to clone the Micron Technology 'Xceed Gray-Scale 30' (part no. XCD-900003) adapter kit, which is designed specifically for the SE/30 in conjunction with one of the above Micron PDS video cards. It provides up to 256 shades of gray scale display on the SE/30's internal display.

The GrayScale adaptor consists of a CRT video card and a cable harness.

Download (the optional) software for [Micron Color 30HR™](#) PDS video card. [Image](#) of Mark Janello's Micron Color 30HR™.

**INDEX:****VIDEO CARD:**[SCHEMATIC](#), (Apple & Micron)[PARTS LIST](#)[LAYOUT](#), (prototype) 6/24/01[ART MASTER](#), (prototype) 6/24/01**HARNESS:**[SCHEMATIC](#)[PARTS LIST](#)[WIRE LIST](#)**MISC:**[COMPONENT DATA](#)[COMPATIBILITY](#)[MICRON on EBAY](#)**PARTS LIST, VIDEO CARD**

QTY	REFDES	PART NO.	DESCRIPTION	notes	from old card?
-			Schematic diagram		
1			Printed wiring board		\$? ??
3	C1,2,3		Cap, ceramic, 0.01uF 1KVDC.....		Y
1	C4	SEK100M050ST	Cap, elctlt, 10uF, 50VDC, 105°C, 5mm.....		\$0.35
1	C5		Cap, elctlt, 47uF, 10VDC, 105°C, 5mm.....		\$0.11
1	C6		Cap, ?????, 0.1uF 100VDC.....		Y
1	C7		Cap, ceramic, 0.1uF 25VDC.....		Y
1	C8	SEK101M016ST	Cap, elctlt, 100uF, 16VDC, 105°C, 6.3mm OD.....		\$0.08
1	C9	SEK101M063ST	Cap, elctlt, 100uF, 63VDC, 105°C, 10mm OD.....		\$0.29
3	D1,2,3	1N914A	Diode,.....		\$0.21
1	J1	39-29-1088	Conn, 8 pin, r/a	Molex 5569 series.....	\$1.58
2	L1,2	IMS-5SWD-65	Inductor, 27 uH 10%	Vishay-Dale.....	Y
2	Q1,2	2N5550	Transistor, NPN.....		\$0.66
1	Q3	PN2222A	Transistor, NPN		\$0.12
2	R1,2		Res, 1.0K ohms, 2W 5%	brn-blk-red-gold	\$0.10
1	R3		Res, 75 ohms, 1/4W 5%	vio-grn-blk-gold	\$0.05
1	R4		Res, 300 ohms, 1/4W 5%	orange-blk-brn-gold	\$0.05
1	R5		Res, 56 ohms, 1/4W 5%	grn-blu-brn-gold	\$0.05
1	R6		Res, 200 ohms, 1/4W 5%	red-blk-brn-gold	\$0.05
1	R7		Res, 12K ohms, 1/4W 5%	brn-red-orange-gold	\$0.05
1	R8		Res, 1.5K ohms, 1/4W 5%	brn-grn-red-gold	\$0.05
3	R9,10,11		Res, 100K ohms, 1/2W 5%	brn-blk-yel-gold.....	\$0.15
1	RV1		Res, var, 200 ohms		\$0.75
1	V1	APCO EIA-1210	Socket, CRT, 7 pin miniature, PC mount.....		Y

**PARTS LIST, HARNESS**

QTY	REFDES	PART NO.	DESCRIPTION	notes	from old card?
-			Schematic diagram		
2	J1,2	39-01-2140	Conn, 14 pin,	Molex 5557 series.....	\$1.54
1	J3	39-01-2120	Conn, 12 pin,	Molex 5557 series.....	\$0.60
1	J4	39-01-2100	Conn, 10 pin,	Molex 5557 series.....	\$0.56
1	J5	39-01-2080	Conn, 8 pin,	Molex 5557 series.....	\$0.50
1	Z1	2643665702	Shield, cable, 1 1/8x11/16x3/8 1.125x.687x.375	Fair-Rite,..... (or Ferroxcube CST17/9.5/29-4S2 or Amidon FB-(43)-6873)	\$0.40
2	Z2,3	2643540002	Shield, cable, 1 1/8x9/16x1/4 1.125x.562x.25	Fair-Rite,..... (or Ferroxcube CST14/6.4/29-4S2 or Amidon FB-(43)-5621)	\$0.64
58		39-00-0039	Terminal, crimp, fem, Terminal, lug Wire	Molex 5556 series.....	\$6.96
2					<b>TOTAL = \$15.90</b>

WIRE LIST, HARNESS				
FROM	Color	TO	Length	Notes
J1- 1	BLK	J2- 1	11"	
J1- 2	BLK	J2- 2	11"	twisted w/WHT (J2- 9)
J1- 9	WHT	J2- 9	11"	twisted w/BLK (J2- 2)
J1- 3	BLK	J4- 5	9"	twisted w/RED (J1-10)
J1-10	RED	J4- 1	9"	twisted w/BLK (J1- 3)
J1- 4	BLK	J2- 4	11"	
J1- 5	BLK	J2- 5	11"	
J1- 6	BLU	J2- 6	11"	
J1- 7	GRN	J2- 7	11"	
J1- 8	BLK	J2- 8	11"	
J1-11	GRY	J4- 7	9"	pass thru two chokes, big and small
J1-12	ORN	J2-12	11"	
J1-13	ORN	J2-13	11"	
J1-14	YEL	J2-14	11"	
J2- 3	BLK	J4- 3	11"	twisted w/RED (J2-10)
J2-10	RED	J4- 2	11"	twisted w/BLK (J2- 3)
J2-11	GRY	J4- 6	11"	pass thru choke w/tp BLK/YEL from J3
J3- 1	ORN	J5- 7	8"	30WH!
J3- 2	BRN	J5- 4	8"	
J3- 3	BLK	J4- 4		twisted w/YEL (J3- 8)
J3- 8	YEL	J4- 8		twisted w/BLK (J3- 3)
J3- 7	GRN	J5- 5	8"	
J3-10	BLU	J5- 1	8"	
J3-12	RED	J5- 2	8"	
J4-09	YEL	J5- 6	7"	twisted w/BLK (J4-10)
J4-10	BLK	J5- 3	7"	twisted w/YEL (J4-09)
J5- 8	BLK		6"	lug, to SE/30 frame.
J5- 8	BLK		12"	lug, to CRT frame

## COMPONENT DATA

CRT-tube socket (won't fit), More sockets

Molex connectors: video card (pdf file), harness (pdf file), harness (pins) (pdf file)

inductor (pdf file)

1N914A

2N5550

PN2222A

Harness chokes: Ferroxcube (pdf file), Fair-Rite, Amidon

SE & SE/30 Schematics

## COMPATIBILITY

The following is an edited version of Mark Janello's post to the CompactMacs list:

"The adapter card is called the 'Gray-Scale 30' (part no. XCD-900003) and is designed specifically for the SE/30 in conjunction with one of these three video cards; it's explicit that it will only work with these:

Xceed Color 30™

Color 30HR™

MacroColor 30™

Provides 1,2,4 or 8-bits per pixel gray-scale display (when no external monitor is being used) My card seems to be a Color 30HR (XCD-900048) and has two video connectors on it, one that attaches to the external monitor plug on the back of the computer (a 12 pin plug), and the one for the optional greyscale card, a 10-pin socket with a plastic plug protecting 10 male pins. The socket is quite like the one for power on an SE/30 motherboard--each pin in its own little plastic box.

My PDS card doesn't have a part number or say 'Color 30HR' on it (I have the original documentation and warranty card that tell me the model) so advice for would-be shoppers is to look for the two sockets on the card. If you hold the card with the PDS connector on the bottom then over on the left are the two sockets.

Mark"

---

**MICRON on EBAY**

[PDS slot video card](#)

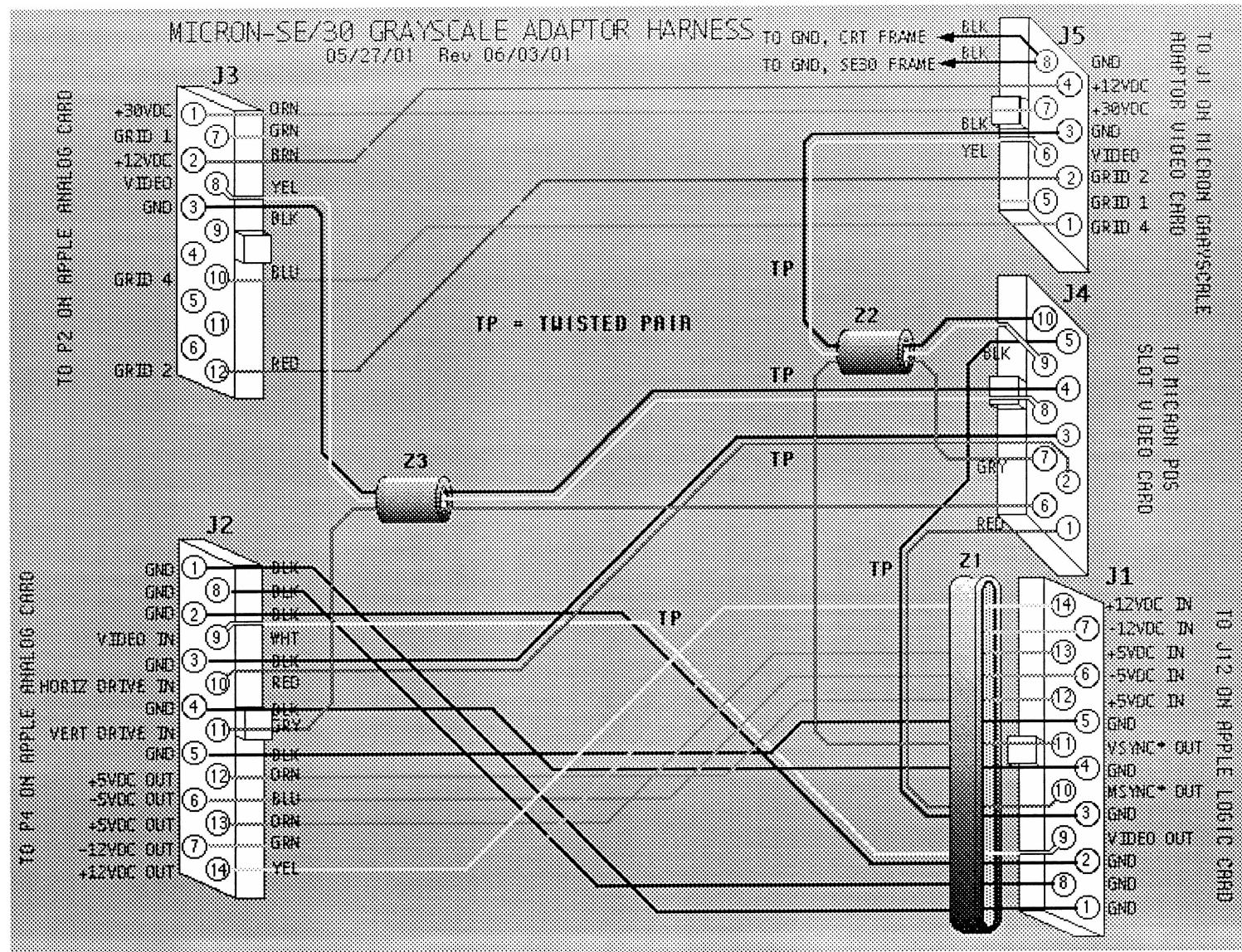
[MacroColor 30 PDS slot video card](#)

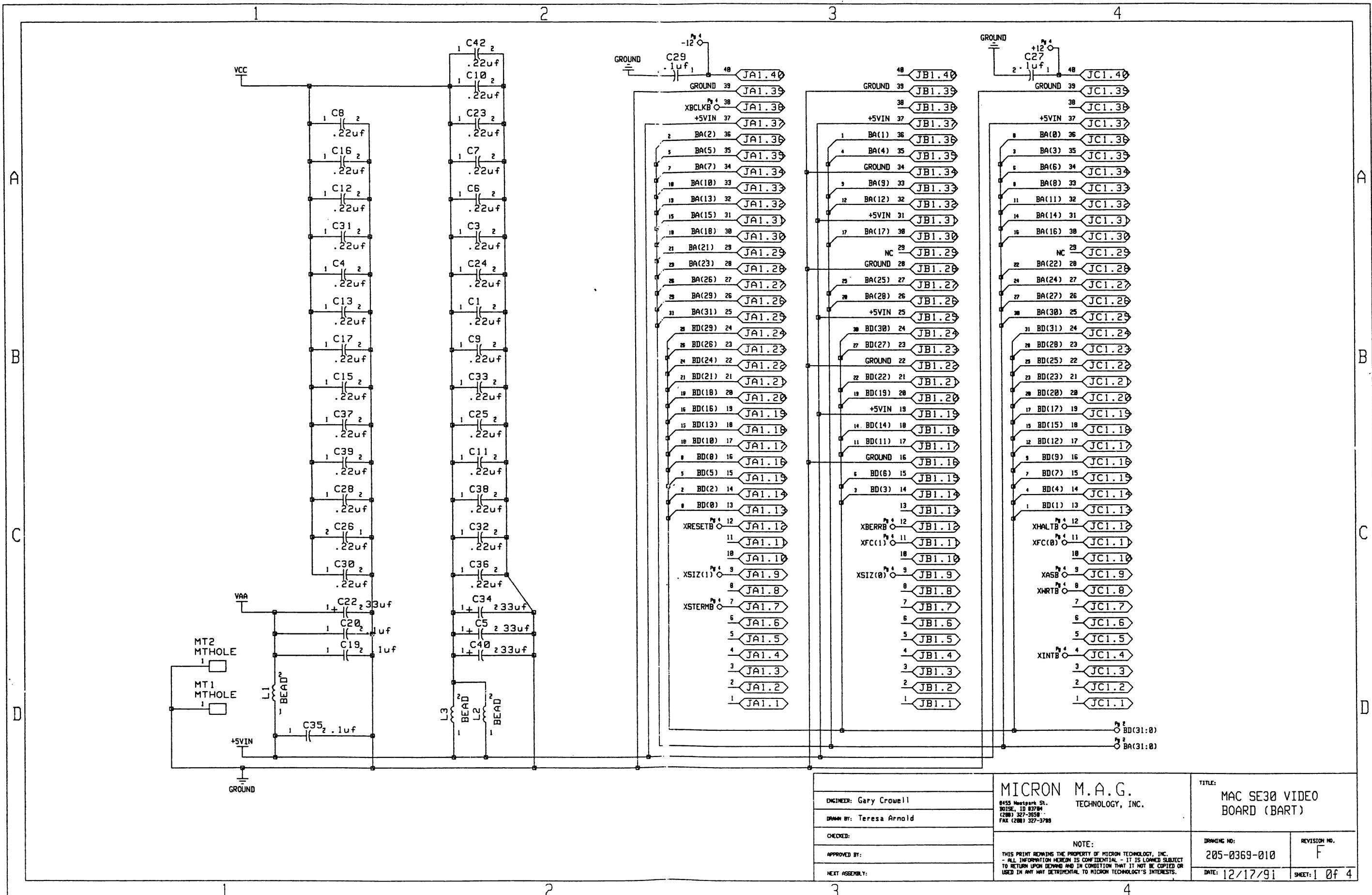
[GrayScale kit & Color board](#)

[PDS slot video card](#)

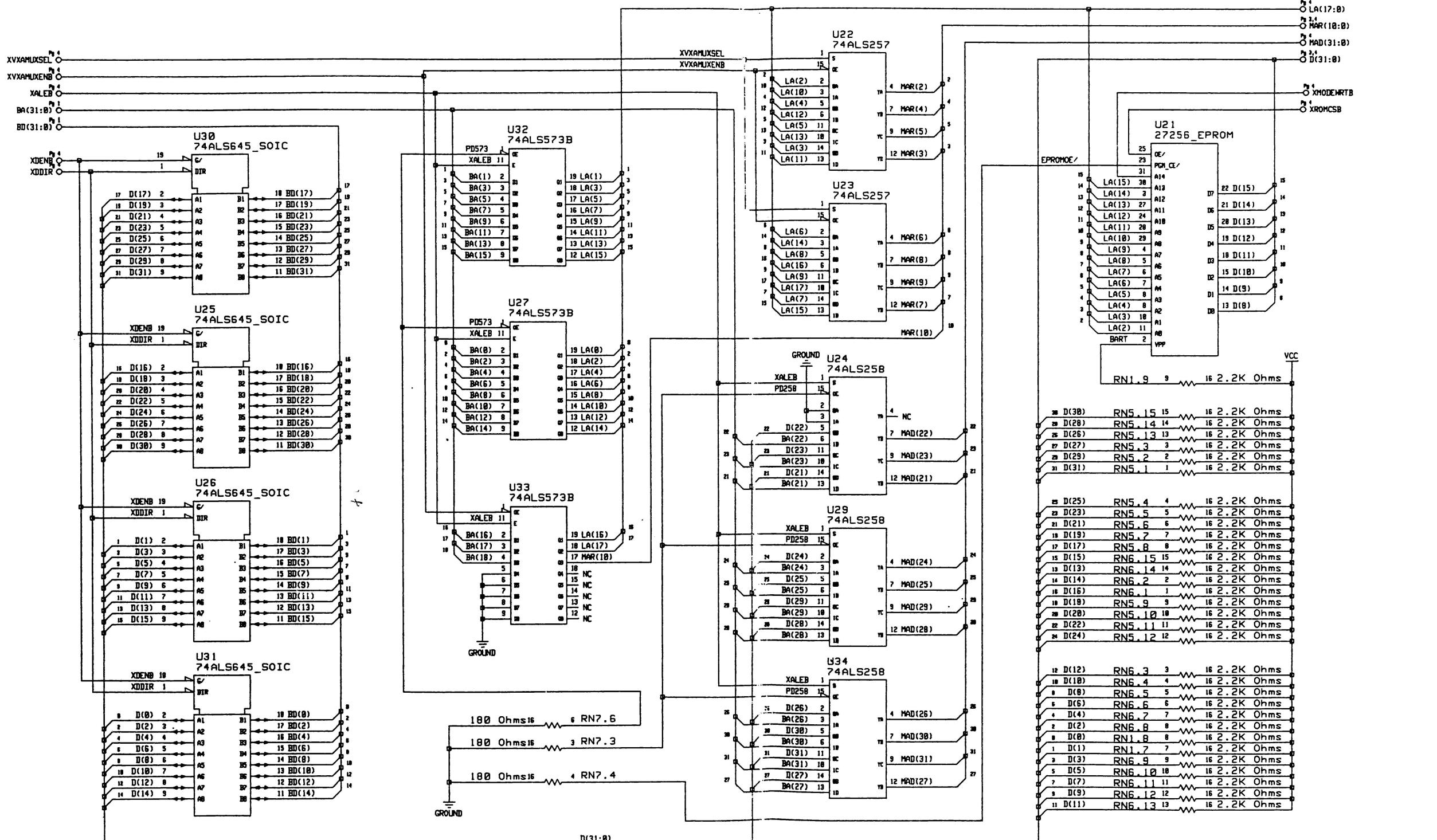
---

This page copyright © 2001 by Gamba. All rights reserved.



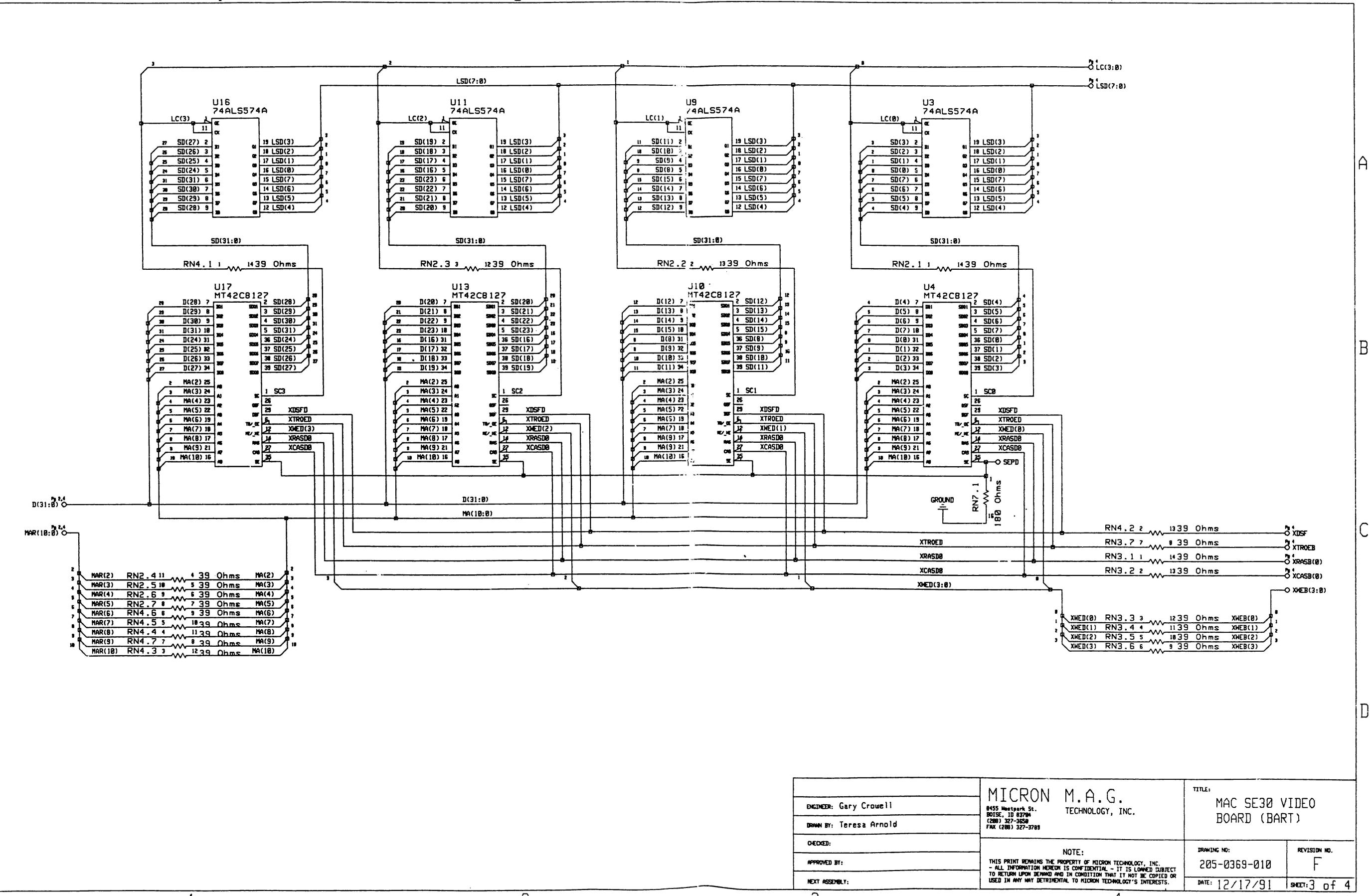


ENGINEER: Gary Crowell DRAWN BY: Teresa Arnold CHECKED: APPROVED BY: NEXT ASSEMBLY:	MICRON M.A.G. TECHNOLOGY, INC. 6455 Westpark St. Boise, ID 83716 (208) 327-3658 FAX (208) 327-3783	TITLE: MAC SE30 VIDEO BOARD (BART)	
NOTE: THIS PRINT REMAINS THE PROPERTY OF MICRON TECHNOLOGY, INC. - ALL INFORMATION HEREON IS CONFIDENTIAL - IT IS LOANED SUBJECT TO RETURN UPON DEMAND AND IN CONDITION THAT IT NOT BE COPIED OR USED IN ANY WAY DETERMINAL TO MICRON TECHNOLOGY'S INTERESTS.			DRAWING NO: 205-0369-010 REVISION NO: F DATE: 12/17/91 SHEET: 1 Of 4



ENGINEER: Gary Crowell	MICRON M.A.G. TECHNOLOGY, INC.	
DRWNR BY: Teresa Arnold	TITLE: MAC SE30 VIDEO BOARD (BART)	
CHECKED:	DRAWING NO: 205-0369-010	
APPROVED BY:	REVISION NO: F	
NEXT ASSEMBLY:	DATE: 12/17/91	

THIS PRINT REMAINS THE PROPERTY OF MICRON TECHNOLOGY, INC.  
- ALL INFORMATION HEREON IS CONFIDENTIAL - IT IS LOANED SUBJECT  
TO RETURN UPON DEMAND AND IN CONDITION THAT IT NOT BE COPIED OR  
USED IN ANY WAY DEDRIMENTAL TO MICRON TECHNOLOGY'S INTERESTS.



ENGINEER: Gary Crowell	MICRON M.A.G. TECHNOLOGY, INC.	TITLE: MAC SE30 VIDEO BOARD (BART)
DRAWN BY: Teresa Arnold	0455 Westpark St. BOISE, ID 83704 (208) 327-3858 FAX (208) 327-3789	
CHECKED:	NOTE:	
APPROVED BY:	THIS PRINT REMAINS THE PROPERTY OF MICRON TECHNOLOGY, INC. - ALL INFORMATION HEREON IS CONFIDENTIAL - IT IS LOANED SUBJECT TO RETURN UPON DEMAND AND IN CONDITION THAT IT NOT BE COPIED OR USED IN ANY WAY DENTRIMENTAL TO MICRON TECHNOLOGY'S INTERESTS.	
NEXT ASSEMBLY:	DRAWING NO: 205-0369-010	REVISION NO: F
	DATE: 12/17/91 SHEET: 3 of 4	

